2

Technical Document 552
October 1989

KAPSE Interface Team (KIT) Public Report

Volume VIII
(Part 1 of 2)

D. L. Hayward

Prepared for the Ada Joint Program Office

Approved for public release; distribution is unlimited.

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed by the Computer Systems Software and Technology Branch, Code 411, of the Naval Ocean Systems Center, San Diego, CA, for the Ada Joint Program Office, Pentagon, 1211 S. Fern Street, Washington, DC 20301-3081 and represents evolving ideas and progress of the KAPSE Interface Team (KIT).

Released by
R. A. Wasilausky, Head
Computer Systems Software
and Technology Branch

Under authority of
A. G. Justice, Head
Information Processing and
Displaying Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
October 1989

3. REPORT TYPE AND DATES COVERED
Final May 1985 to October 1985

4. TITLE AND SUBTITLE
KAPSE INTERFACE TEAM (KIT) PUBLIC REPORT
Volume VIII (Part 1 of 2)

5. FUNDING NUMBERS
C:
PE: 0603226F
PR:
WU: DN288 534

6. AUTHOR(S)
D. L. Hayward

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NOSC TD 552

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Ada Joint Program Office
Pentagon
1211 S. Fern Street
Washington, DC 20301-3081

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report is the eighth in a series and represents evolving ideas and progress of the KAPSE Interface Team (KIT).

14. SUBJECT TERMS
software engineering
CAIS Ada
APSE interface standard
programming languages

15. NUMBER OF PAGES
476

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT

CONTENTS

CONTENTS (continued)

Section 1

INTRODUCTION

INTRODUCTION

This report is the eighth in a series that is being published by the KAPSE Interface Team (KIT). The previous reports are as follows:

Vol. #	NOSC Report #	Date	NTIS Order #
I	TD-209	4/82	AD A115 590
II	TD-552	10/82	AD A123 136
III	"	10/83	AD A141 576
IV	"	4/84	AD A147 648
V	"	8/85	AD A160 355
VI	"	TBD	TBD
VII	"	TBD	TBD

This series of reports serves to record the activities which have taken place to date and to submit for public review the products that have resulted. The reports are issued to cover approximate six-month periods. They should be viewed as snapshots of the progress of the KIT and its companion team, the KAPSE Interface Team from Industry and Academia (KITIA); everything that is ready for public review at a given time is included. These reports represent evolving ideas, so the contents should not be taken as fixed or final.

MEETINGS

During this reporting period (November 1985 through October 1986) the teams met in January 1986 in San Diego, CA, in April 1986 in Atlanta, GA, in July 1986 in San Francisco, CA, and in September 1986 in Minneapolis, MN. The approved minutes from these meetings are included in this report. Also included are the minutes from the April COMPWG meeting.

The CAIS Implementors Group (CIG) has continued to meet. It has now become a Working Group under the Environment Committee of SIGAda and so meets at every SIGAda meeting. The minutes of the CIG's July 1986 meeting are included in this report.

PROGRAM MANAGEMENT

This report includes the 1986 Management Plan for the CAIS effort.

COMMON APSE INTERFACE SET (CAIS)

The Proposed MIL-STD-CAIS (dated January 1985) was sent out to the DoD community for formal standardization review during this reporting period. Over 600 comments were received, answered and responded to in the form of changes to the text of the document. On 9 October 1986 the CAIS Standardization Working Group gave their approval to the answers and proposed changes, thus approving the document as a standard. It is now known as DOD-STD-1838. During the next reporting period, the proposed changes will be implemented and the document will be formally turned over to Navy Publications for production as a military standard.

Competitive procurement of a contractor for CAIS Revision A, or CAIS-A for short (previously known as CAIS Version 2), has resulted in the award of the contract to SofTech, Inc. in December 1985. It is a three-year contract which is intended to produce the CAIS-A specification, a prototype, and several accompanying documents, including a rationale. Some of the results of their work are included in this report.

REQUIREMENTS AND DESIGN CRITERIA (RAC)

The RAC and its rationale were completed during this reporting period. It was especially significant to have the new CAIS-A contractor on-board during this period. As the ones tasked with fulfilling the requirements in the RAC, they brought out many heipful comments and questions which helped to direct the finalization of the RAC. The RAC was published as NOSC TD 1121 and is now available from NTIS; ask for order number AD A184 488. This version of the RAC is included here.

As an initial exercise in preparation for CAIS-A, a comparison of DOD- STD-1838 (CAIS-1) was made to the RAC. Although the RAC provides requirements for CAIS-A, not 1838, and 1838 was in no way expected to measure up to all of the requirements in the RAC, CAIS-A is required to be upwardly compatible with 1838 while fulfilling the requirements of the RAC. For this reason, it is of interest to determine just how far away the starting point is from the goal. The results of this study are documented here in the RAC/CAIS Version 1 Compliance Study.

Another debate in the preparation of the RAC has been the best nature of the entity management system. Several speakers were invited to KIT meetings during this reporting period to provide some information regarding object- oriented data models. Papers by one of these speakers (Thomas Atwood) are included in this report.

Other papers and briefings relevant to the RAC are also included. The issues covered include security (by Compusec, subcontractor to SofTech on the CAIS-A contract), distribution (by LeGrand), node model granularity (by Rogers), related NASA Space Station requirements (one by Chevers, the other by McKay), and ALS/N (by NAVSEA PMS-408).

COMPWG

The Compliance Working Group has finished some of its work regarding approaches to semantic description of the CAIS. This work is reported here in papers by Freedman and by Lindquist, et al. New work by the COMPWG is beginning to look at other aspects of compliance, including quality assurance determination, as represented here by the paper by Stiles.

GACWG

The Guidelines and Conventions Working Group has included in this report a survey regarding interoperability problems. The purpose of this survey is to gather inputs for an interoperability guide which the GACWG will generate, along the same lines as the transportability guide already submitted.

DEFWG

The Definitions Working Group has submitted an updated KIT/KITIA Glossary during this reporting period, and it is included in this report. The objective is to keep a clear glossary of terms used in the documents in order to provide consistent definitions for use by the various authors. The first priority of the DEFWG is to identify terms used especially ones used in more than one document. The next is to determine whether or not these terms are being used consistently and, if not. to try to resolve the inconsistencies.

PROTOTYPES

Reports on several prototyping activities are available in this report. The first, in the form of both a paper and a briefing, is from the IBM prototyping team, by Vermette. The second is from the Gould prototyping team, by Carr, et al. The last is an update on the TRW prototype.

RELATED EFFORTS

A number of efforts of interest to the KIT/KITIA are beginning to emerge , and members of the teams are tracking them and reporting on them. Two such reports are included here. The first discusses AT&T's new System V Interface Definition (SVID) and its relationship to the CAIS. The second consists of a briefing and two reports on "PCTE: A Basis for a Portable Common Tool Environment". The briefing was one of several on PCTE presented during the last few KIT meetings. The first report discusses the organization and design guidelines of the PCTE project. The second report compares PCTE to the RAC.

CONCLUSION

This Public Report is provided by the KIT and KITIA to solicit comments and feedback from those who do not regularly participate on either of the teams. Comments on this and all previous reports are encouraged. They should be addressed to:

Duston Hayward
Code 411
Naval Ocean Systems Center
San Diego, CA 92152-5000

or sent via ARPANET/MILNET to HAYWARD@NOSC-TECR.ARPA.

Section 2

TEAM PROCEEDINGS

KIT/KITIA MINUTES
MEETING OF 14-16 JANUARY 1986
SAN DIEGO, CALIFORNIA

AGENDA: SEE APPENDIX A

ATTENDEES: SEE APPENDIX B

MEETING HANDOUTS: SEE APPENDIX C

14 JANUARY 1986

1. OPENING REMARKS

- Hans Mumm, Acting KIT Chairperson, brought the meeting to order.

- New KIT/KITIA members and visitors were introduced. Mike Tedd of the University of Wales represented the United Kingdom on the KIT for Sue Bond, who will be the primary representative. Bill Wood was present representing the Software Engineering Institute. Jean Tardy represented the Canadian National Defense Headquarters. Dit Morse, Oracle Corporation, has been added to the KIT membership. Charlotte Winnick is the new KITIA representative for Norden Systems. Esa Nurmi is the new alternate KITIA representative for Oy Softplan AB, Finland. Steve Huseth, Honeywell/SRC, is replacing Mike Kamrad on the KITIA. Dr. Roy Freedman, New York Polytechnic University, represents Hazeltine corporation on the KITIA. New additions to the KITIA include Dr. Tim Lindquist, Arizona State University, Fernando Gallo of Bull Corporation, France, and Tim Harrison, Texas Instruments. Visitors at this meeting included Ed Chevers, NASA, Houston, Marshall Lee, Gunter Air Force Base, Brian Close, British Defence Staff, Dave Andrews, Andyne Computing Limited, Canada, Steve Roski, Logicon, Bob Stevenson, Gould Corporation, Brian Schaar of Techplan, Les Anderson and David Collom of NOSC.

2. GENERAL BUSINESS

- Jinny Castor is now the permanent Director (vice "acting) of the Ada Joint Program Office.

- Sue LeGrand has left Ford Aerospace and is now at SofTech, Houston.

- This meeting was intended to provide a technical orientation for the CAIS Version 2 Design contractor. As a result, the attendence at this meeting was larger than normal to provide the various participants in the Version 2 design effort exposure to the technical issues identified by the KIT/KITIA during the formulation of the Requirements and Design (RAC) document and the CAIS Version 1 document.

- The CAIS Version 2 Design contract from the Naval Ocean Systems Center has been awarded to SofTech Inc, and their subcontractor, Compusec. The CAIS Version 2 effort has been segmented by SofTech into three functional groups: a Design Team based in Waltham, a Prototype Team in San Diego, and a Security Team from Compusec in San Diego. Gary Pritchett is the Contract Manager and Rich Thall will be the Technical Director. Representatives from the SofTech design team attending this meeting include from the Waltham office, Rich Thall and Mark Conway, from the San Diego office Gary Pritchett, Geoff Clow, Tom Robinson, Wally Nidzieko, and Ron Santina. Representatives from Compusec in San Diego include Bobby Miller, Jim Perry, George Stones, Alexander Enzmanh, Bill Townsend, and Russ Burke.

- The KIT Support contract, also from NOSC, was awarded 31 December 1985 to TRW Defense Systems Group of Redondo Beach, California. Hal Hart is the project manager and has organized two groups to support the Naval Ocean Systems Center. The Space Park team supports the CAIS Version 1 prototyping actiivities, Stoneman and Requirements and Criteria document generation and a San Diego group to provide the general KIT support including the CAIS to RAC traceability analysis and MILNET support. Representatives from Redondo Beach in attendence included Hal Hart, Frank Belz, Frank Tadman, Tony Alden and Judy Kerner. San Diego participants were Jack Foidl, Ann Evans and DeWayne McCracken.

- A CAIS Prototype contract has been awarded from the Canadian National Defense Headquaters to Anydyne Computing Limited of Canada.

3. AJPO COMMENTS

- LCDR Philip Myers, Navy Deputy Ada Joint Program Office, welcomed Professor Mike Tedd representing the United Kingdom Ministry of Defence (MOD) to the KIT.

- Brian Close is the U.K. MOD liaison representative in Washington, D.C..

- All KIT and KITIA members are requested to report any problems they are experiencing with their MILNET accounts to the KIT or KITIA chairperson and not directly to the AJPO.

- Regarding the question of export controls, the CAIS has not been raised as an issue area for export control of technology.

4. KITIA CHAIRMAN REPORT

Herm Fischer, KITIA chairperson reviewed the status of KITIA activities:

- The KITIA views project, designed to provide a template of criteria to evaluate CAIS on economic and technical criteria, has been completed. It was formulated to help companies evaluate environments.

- STARS staff has been showing increased attention in the development of the CAIS and its progress.

- KITIA members supported a CAISWG meeting at Carnigie Mellon involving representatives from the Software Engineering Institute.

- New KITIA approved membership will be announced after the KITIA Executive Committee meeting.

5. EVALUATION AND VALIDATION TEAM REPORT

Ray Szymanski, E&V Chairperson, presented a status report of the E&V activities:

- Contracts - The Analytical Sciences Corporation (TASC) are continuing work on the E&V Configuration Management Plan, definition of an E&V Classification Schema, and an accompaning Reference Manual and Guidebook.

- A set of the Ada Compilier Prototype Benchmarks test suite can be obtained via a written request to the following address. A Users Manual is included with the test suite which has a report writer available for execution in a batch mode. Requests should be directed to:

  Softech Inc.
  Presidential Drive
  Fairborn, Ohio

  Attn: Compiler Evaluation Benchmarks

- CAIS Validation Capability - there were no bidders on this October 1985 RFP which is now scheduled for re-release in mid-February 1986.

- Ada Compiler Evaluation Capability - The CBD for the RFP is expected mid-April.

- CAIS Operational Definition Work - The procurement details for this activity have almost been resolved. Dr. T. Lindquist and his team are expected to be contiuing this work in the near future.

- The E&V Team is scheduled to present briefings at the SIGAda Meeting in Los Angeles at the end of February including support to a possible Birds of a Feather meeting on Thursday evening 2/27/86. Planned is a presentation on E&V documents such as the APSE Evaluation document, the Validation Procedures document, Reference Manual and Guidebook.

- The E&V Team wants to find kindred spirits in the E&V area and possible new members on the E&V team. The E&V Working Group chairs have requested additional help to supprt their activities.

6. WORKING GROUP REPORTS

- RACWG - Hal Hart reported that there is a new copy of the Requirements and Criteria document dated 9/13/85 which has minor changes from the July version. The offical RAC comment review period is open following the public presentation of the document at the SIGAda in Boston. Frank Belz attended the Ada UK meeting and should provide some feedback later in this meeting on the European comments on the document. 15 March 86 is the established cutoff date for submission of comments. The RAC is to be mailed out to all those who registered for the CAIS MIL-STD-1 mailing list.

- STONEWG - Ann Reedy reported the STONEWG held an interim meeting in Phoenix in November. Their plan is to assemble their completed sections and publish a rough draft for Kit/Kitia review.

- COMPWG - Bernie Abrams gave a status report of the COMPWG activities since the last KIT/KITIA meeting. They are preparing a paper on methods of specifying formal semantics including operational, denotational, axiomatic and also examining some potential testing methods for testing CAIS compliance. Lloyd Stiles is preparing a methodology for QA analysis. Jack Foidl and Guy Taylor will now be continuing the traceability analysis for the CAIS and the RAC. A draft report on this work is planned for the April meeting to solicit KIT/KITIA comments. Bernis Abrams will work on formulating a Test Methodology for the CAIS.

- GACWG - Ron Johnson reported the GACWG is starting to transition from generation of the Tool Transportability Guide to turn over the document for finalization and their starting work on an Interoperability Guide. The GACWG goal is a draft planned for the April meeting and would appreciate any help other KIT/KITIA members could provide.

- CAISWG - Clyde Roby presented the progress of the CAISWG. During this quarter the CAISWG has generated a draft CAIS Rationale, a draft CAIS Readers Guide, conducted a review of the submitted CAIS comments and prepared responses and supported a CAIS Workshop attended by members of the CAISWG, the CAIS Implementors Group and the Software Engineering Institute. The CAISWG is expecting to continue work on the Rationale document and support the CAIS Standardization Process. At the April meeting the CAISWG plans to have a final CAIS Readers Guide and the proposed changes to the MIL-STD-CAIS Version 1.

- DEFWG - Hans Mumm reported that the DEFWG has an up to date Glossary on-line at KIT-INFORMATION <password KIT>. Some hard copies were available at the meeting.

7. GENERAL ANNOUNCEMENTS

- Some future meetings that may be of interest to KIT/KITIA members include the SIGAda in February in Los Angeles during which there will be a CAIS Review and an E&V Status Review. A CAIS Implementors Group meeting is also scheduled. A Conference on Ada Techology is being sponsored by the Army CECOM in March in Atlanta.

- KIT/KITIA members were encouraged to use "Ada2Ø" as their NET address since there will probably be a move from the ISIF machine in about a year. Also note Tim Lyons is now TLYONS on the NET.

- Jack Foidl, as KIT Support Contractor, reminded visitors that California law now requires use of seat belts as of 1 January 1986. The NOSC supply of CAIS documents is now exhausted. Requests for additional copies should now go to the AJPO which has an additional supply. A mailing will be conducted shortly to supply those people who were on the CAIS mailing list with a copy of the RAC as part of the Public Review process. Doug Wrege is finalizing the arrangement for the April meeting in Atlanta and has negotiated a special rate for government personnel. Hans Mumm was congradulated for doing a super job making the arrangements for this meeting in the absence of any support.

- Hans Mumm then presented the status and plans for the CAIS Standardization Process. During November 1985 CAIS Version 1 was distributed to the 3 services and to 8 industrial organizations for formal review. SIGAda has requested an extension of the review period to March 1986 (offical USAF date is 1Ø March 1986). During April the received comments will be distributed to the CAISWG. The CAISWG will conduct a review of the comments and provide their recommendations to the CAIS Control Board in June 1986. In July the CAIS Control Board will either vote for standardization forwarding the CAIS for approval or identify changes to be made in the document.

- Bob Stevenson (Gould) would like to know if any members of the KIT/KITIA would be interested in obtaining copies of their CAIS Prototype. The goal would be to accelerate CAIS activities through utilization of the GOULD developed prototype. Contact Bob directly for additional information.

8. MORNING BREAK

9. SPACE STATION PRESENTATION

- Ed Chevers presented an overview of Ada activitiies related to the NASA Space Station Program. NASA, with support of the University of Houston at Clear Lake, is developing a distributed test bed system connected by fiber optic link representing a series of nodes with specific functionality. They needed to write a distributed operating system (fault tolerant, security requirements) in Ada. NASA has established an Ada Technology Operating Plan (ATOPs) which is a series of tasks to be developed for which NASA provides development facilities via this test bed. The regional offices and the Space Station Program Office have recommended Ada as the baseline language for Space Station flight applications software excluding the operating system, dbms, user interface and ground control/distribution. It is expected that these present exclusions will be removed in about 6 weeks. NASA now has 65 ATOPs (including 3 from Europe) defined and 1Ø Ada software systems operational on the JSC/UH network. Specification for a Software Supprt Environment (SSE) will be issued in April 86 with contract award in the November/December 86 period. The SSE will support all Space Station software. The Space Station expects the initial 2-3 million lines of code (LOC) to expand to 10-13 million LOC on space station plus huge amounts of ground support software. The SSE is viewed as just another node in the Space Station management network. The SSE operating system is primarily software development oriented whereas other nodes are akin to real time process controllers. Thus it appears that the SSE host should support the "Virtual Ada Machine" concept proposed by McKay and implications of this concept are that the Ada Run Time Support Environment (ARTSE) and Common APSE Interface Set (CAIS) are not independent functions. NASA expressed concern that run-time support environment is not in CAIS Version 1. NASA forsees a strong requirement for layered software with standard interfaces and would like to get NASA people tied in with KIT/KITIA Interoperability work.

10. LUNCH BREAK

11. CAIS VERSION 2 PRESENTATION

Rich Thall presented an overview of the CAIS Version 2 Design contract awarded by NOSC to the SofTech/Compusec team in December 1985. The base contract is scheduled to be completed in 18 months with one additional 18 month option.

- The Version 2 Standard will be designed to meet practical requirements and will be supported by a prototype, a Rationale document, an Implementors Guide, and Formal Semantics. This development is planned to be responsive to the Public Review inputs.

- The team is divided into three main groups headed by G. Pritchett. The Design team is based in Waltham under R. Thall, the Prototype team in San Diego under G. Pritchett, and the Security team from Compusec in San Diego led by J. Perry. Compusec is a consultant to SofTech to provide expertise in the security area.

- The major contract deliverables include the CAIS 2 Standard, a prototype of CAIS Version 1 which will evolve to Version 2 as a tool to try out ideas for the Version 2 design, a Rationale document, an Implementors' Guide, useable formal semantics and Issue Reports which will address various issues as they are identified. NOSC will have to decide if there will be interim releases of the developing prototypes. The SofTech proposal was to modify the ALS KAPSE for the CAIS prototype on a VAX/VMS system. SofTech would like to develop a toolset and move it to different prototypes; the ALS toolset could be a candidate for consideration. As the CAIS 2 is developed, the teams will be maintaining a Rationale Log for quick release of a Rationale document to support Version 2. Additional support will be provided for Public Reviews, KIT/KITIA meetings and CAISWG meetings.

- The current schedule shows a Draft CAIS 2 due late December 1986 with a final document one year later following the Public Review process. Major changes to the draft will not occur during the Public Review cycle. The initial CAIS 1 prototype is due in the first quarter of CY-87 and planned to evolve to a CAIS 2 prototype a year later. The draft Rationale follows the CAIS 2 document by about a month with a final due a month after CAIS 2 is finalized. The Implementors' Guide is expected mid-1988 and the formal semantics about three months later. Since the work is performed under government contract it is expected that the prototypes will be available in the public domain, as is the MITRE prototype.

- The major activities to be performed under the current Delivery Order include the CAIS 1 prototype design, an analysis of the RAC for CAIS 2 to identify differences from CAIS 1, and support to the KIT/KITIA meeting in San Diego and the CAIS Implementors' Group (CIG) at the Los Angeles SIGAda. The CAIS 1 prototype is intended to be functionally complete in so far as the hard areas are included such as access control, the node model, etc. The initial prototype may not, however, support all available terminals.

- A discussion of implementing security requirements followed. If the plan is to "hack" the ALS on VMS, how can you demonstrate mandatory access when you are using an operating system that can't support this performance. A re-host would require re-writing the system dependent parts. If an existing Trusted Computing Base (TCB) is utilized for the prototype it is not clear it would reflect the technology required for the CAIS. It may not be realistic to expect full resolution of the security issues in the time frame allowed. The prototype is intended to provide a basis for further analysis and further definition of issues such as this.

- A clarification of a previous recommendation by SofTech to dump the CAIS and use the ALS KAPSE was made. Rich Thall indicated he proposed two standards: ALS as the standard for piggyback implementation and CAIS as the standard for bare machine implementation based on state-of-the-art technology in software engineering. Rich feels that piggybacking requires the lowest common denominator and that CAIS 1 or even CAIS 2 will not be appropriate for piggybacking. Nor does he think there is an operating system available today to support an efficient piggyback implementation.

- The purpose of the CAIS 1 prototype is to emphasize design issues for functionality but a key consideration is the performance efficiency of the CAIS 2, which will be based on an analysis of the CAIS 1 prototype. One of the strengths of the CAIS 1 is the unity of design. SofTech intends to construct and maintain a matrix of those items/features which are special cases.

- Regarding "deferred" items from Version 1, the KIT/KITIA assumption was that they would be included in CAIS 2. Rich reported this may not be necessarily true since they may not be compatible with Version 1 or there may be differences between the RAC and CAIS 1 compatability. They will be developing Issue Reports on these items which will be reported back to the KIT/KITIA for discussion. The RAC is expected to evolve during the CAIS 2 design process and these Issue Reports will provide a good basis for coordination with the KIT/KITIA. Compusec is currently working on an issue report addressing the security related areas. Tim Lyons suggested that since the best experts were in the DoD, could they be enlisted to address this area. Philip Myers indicated he could identify individuals to support this activity but be prepared to ask specific questions about requirements and not expect these people to provide interfaces. They are currently reviewing the CAIS 1 and will be providing comments.

- Rich reported the contract kick-off meeting was held 6 December 1985, the Compusec subcontract is in place, and the first NOSC Delivery Order has been executed. Staffing is progressing and the staff is very familiar with ALS interfaces and versed in the issues related to CAIS. An initial review of the RAC is in progress for early April delivery to NOSC.

- The expected tasks related to the CAIS 2 design were presented. These included a study of the RAC requirements with a report reflecting the differences. SofTech plans to meet regularly with the RACWG. CAIS 1 will be studied to develop the CAIS 1 prototype design. A problem list will be constructed from the received comments on CAIS 1 and a special case matrix will be constructed. Other standards will also be examined such as PCTE, IEEE MOSI, UNIX, OSCRL, ISO Network Standard, etc.. A strong area of concern for SofTech is in the user interface area. They would like help to include mice, bit-mapped graphics and what the KIT/KITIA may feel are future I/O devices. There is little standardization in this area and the CAISWG did not have the resources to address this in CAIS 1. SofTech would like to host tools on the CAIS 1 prototype to use this as a basis to try out new ideas for CAIS 2. The results of the preceding activities will be considered in the design of CAIS 2.

- Three main areas are currently targetted as subjects for the Issue Reports; CAIS 1 deferred items, relation between mandatory and discretionary access control and a layered or "pluggable" design concept. This last area may result in quasi-official utility packages like Chapter 14 and may support items such as terminal I/O, for example. Issue Reports are expected to be generated from the KIT/KITIA as well as SofTech.

- SofTech asked the KIT/KITIA for validation of the RAC requirements since CAIS 2 is based on the RAC. For example, if the RAC requires "multi-lingual" support, what are the languages to be supported? They would like to understand the context of the issues to be sure closed issues are explicitly closed rather than implicitly closed. Hal Hart suggested that SofTech review the RAC Rationale which addressed many issues and to use the RAC Comment Form for any additional issues SofTech may have with the current RAC.

12. AFTERNOON BREAK

13. NAMED WORKING GROUP MEETINGS - RACWG MEETS WITH CAIS 2 CONTRACTOR

14. ADJOURN FOR DAY

WEDNESDAY 15 JANUARY 1986

15. ORGANIZE INTO RAC SECTION WORKING GROUPS TO MEET WITH CAIS 2 CONTRACTOR

16. MORNING BREAK

17. RE-CONVENE INTO RAC SECTION WORKING GROUPS

18. LUNCH BREAK

19. RE-CONVENE INTO RAC SECTION WORKING GROUPS

20. AFTERNOON BREAK

21. RAC VOTING

22. ADJOURN KIT/KITIA - KITIA EXECUTIVE MEETING

THURSAY 16 JANUARY 1986

23. COMPWG, GACWG, STONEWG MEET WITH CAIS 2 CONTRACTOR

24. AFTERNOON BREAK

25. PCTE/RAC REPORT

- Herm Fischer reported on how the PCTE conforms to the current RAC. The general impression is that PCTE meets most of the requirements of the RAC except for not being written in Ada and for the security area. Some minor differences included that exact identifiers are not currently in PCTE but it was felt they could be provided. Task waiting could also be added but that would impact the Ada Run Time Environment. Global searches are not fully supported. In summary, PCTE has a similar underlying model to CAIS Version 1 and is RAC compatible. Those who desire a copy of the PCTE to CAIS Version 1 report distributed at Saratoga Springs should contact Herm.

26. NAMED WORKING GROUP REPORTS

- CAISWG - Clyde Roby reported the CAISWG discussed issues related to security and processing. F. Gallo gave the CAISWG a PCTE overview presentation and its relationship to the CAIS development effort.

- STONEWG - Ann Reedy reported the STONEWG is reviewing their written material.

- GACWG - Ron Johnson reported the GACWG has made progress on the outline for their Interoperability Guide. An Interoperability Problem Report was distributed to the KIT/KITIA (see Handouts) for collection of issue areas the KIT/KITIA could help identify. Philip Myers suggested an electronic form be put up under the KIT-INFORMATION <password KIT>. The GACWG is hoping to use this form as a primary means of information gathering. Anyone having knowledge of groups that are (or have) conducting Interoperability studies, such as MITRE, should pass the data to the GACWG.

- COMPWG - Bernie Abrams reported the COMPWG is progressing on the RAC/CAIS traceability analysis and will deliver a report on this activity at the April meeting. A paper on formal semantics will also be published. They are also discussing testing and the problems involved with testing something as large as the CAIS.

- RACWG - Hal Hart reported the RACWG is still revising Section 5 of the Rationale. The RAC will continue to evolve via the Change Proposal process. RAC Change Proposals will be distributed via the NET. They have identified areas for claification and may add a Section 7 for Resource Sharing or other orthogonal issues. The RACWG wanted to make it clear they are aware that there are contradicting requirements in the document. Requirements were included based on their individual merits and may, in fact, contradict other requirements. It is intended that the RAC Comments Form be utilized as a means of identifying these areas for future determination/resolution and the basis for their inclusion will be identified in the RAC Rationale document. Hal will take the lead in generation of the proposed Section 7.

- DEFWG - Hans Mumm reported Judy Kerner is now chairperson for the DEFWG.

27. KITIA REPORT

Herm Fischer, KITIA Chairperson, presented the results of the KITIA meetings held during this joint meeting.

- At the KITIA Executive Committee meeting approval was obtained for membership for F. Gallo from Bull, T. Harrison from Texas Instruments and T. Lindquist from Arizona State University. L. Stein was approved as the full-time representative from the Aerospace Corporation.

- At the KITIA meeting it was decided to focus the KITIA Working Groups on the following areas that map to RAC sections:

WG	Area	RAC Sec.
1	Process Management	5
2	Entity Management	4
3	I/O & User Interface	6
4	Security, Distribution (Knowledge Base (?))	1,2,&3

- Working Group chair elections have been deferred to the next meeting.

- The KITIA plans to schedule an hour dedicated to the KITIA Working Groups to focus on their various areas followed by an hour dedicated KITIA meeting to discuss the results of the Working Group meetings and formulate an Industry/Academia position on various issues (which may differ from a DoD or KIT contractor position).

- The KITIA may invite "tempory attendees" as guest experts to provide meaningful insignt to mainstream discussions.

- Election for the KITIA chairperson was held resulting in the re-election of Herm Fischer as chairperson.

28. KIT/KITIA WRAP-UP SESSION

- Hans Mumm announced that TRW, as support contractor, was to be tasked to finalize the Tool Transportability Guide for the GACWG.

- There were two possible considerations for extension of the KITIA support period, which expires at the end of this meeting. One recommendation is to request extension to the eighteen month segment of the CAIS Version 2 contract and re-evaluate at that time. The second was to request a two-year extension. The Ada Joint Program Office is nervous to request a two-year extension since that represents a fifty percent increase over the original committment requested from the participating companies. NOSC and the AJPO will work this problem.

- Gould will provide a demonstration of their CAIS prototype at the April meeting. A discussion of the value of inviting security "experts" for the next meeting resulted in a request for specific questions to be sent to Hal Hart (HALHART@ADA2Ø) for consolidation for the AJPO. From the list and scope of questions an agenda can be prepared. Herm Fischer noted that Ada-Europe spent a good deal of time trying to identify how security issues affected interfaces. Rich Thall suggested it would be worthwhile for the KIT/KITIA, as a technical advisory body for the CAIS development, to have some background in security to formulate specific questions or evaluate SofTech design decisions. Frank Belz noted it would be valuable to know where Trusted Computing Bases are changing in the future, in software and hardware, since that will impact the direction the CAIS Version 2 must progress.

- At the present time there are presentations planned at the next meeting from Compusec, SofTech, and TRW (including possibly a draft of their version of the Transportability Guide). Dave Pogge will present the Ada concerns the China Lake personnel have. Herm Fischer will give a presentation on User Interfaces. It is expected that CDC will give an ALS/N overview. Ann Reedy requested that presentation time be controlled so ample time is available for Working Groups.

- Hans is to address preparation and distribution of meeting minutes with the support contractor to increase the detail and have them on the NET in a quicker time frame.

- SofTech expressed their appreciation for the support provided by KIT/KITIA to help them fully understand the issues being considered. They would like to set up a liaison with the RACWG due to the strong interrelationship of the RAC/CAIS. Distribution of NET messages to SofTech can be directed as follows:

Account	Recipient
SOFKIT	Waltham Design Team
RTHALL	RAC Comments for SofTech review
GPRITCHETT	San Diego Prototype Team

JPERRY	Compusec Security Team
CAISV2	Mail to all of the above

29. BREAK FOR LUNCH

30. COMPWG/RACWG/STONEWG MEETING WITH V2 CONTRACTOR

31. CAISWG MEETING WITH V2 CONTRACTOR

32. MEETING ADJOURNED

APPENDIX A
AGENDA FOR KIT/KITIA MEETING
13-16 JANUARY 1986
SAN DIEGO, CALIFORNIA

Monday, 13 January:
Named working group meetings as needed.

Tuesday, 14 January:

0800-0830: Arrive and Settle
Coffee, Donuts, & Danish

0830-1030: General Business and Announcements

- Introductions
- Status Reports (Contracts, etc.)
- KITIA Chairman Report
- E & V Report
- DIANA Chairman Report
- XWG Chairmen Reports
- Announcements
- Meeting Shedule
- Local Arrangements
- New Business

1030-1045: Break
Coffee

1045-1145: Space Station Presentation

1145-1315: Lunch

1315-1430: CAIS Version 2 Contractor Report

1430-1445: Break
Coffee and Soft Drinks

1445-1600: Named Working Group (XWG) Meetings
(RACWG meets with CAIS V2 Contractor)

1600-1700: RAC Sections
(RAC Section 6 meets with CAIS V2 Contractor)

Wednesday, 15 January:

0745-0800: Coffee, Donuts, & Danish

0800-0945: RAC Sections
(RAC Section 4 meets with CAIS V2 Contractor)

0945-1000: Break

	Coffee
1000-1130:	RAC Sections (RAC Section 5 meets with CAIS V2 Contractor)
1130-1300:	Lunch
1300-1430:	RAC Voting
1430-1445:	Break Coffee and Soft Drinks
1445-1600:	RAC Voting
1600-1700:	KITIA Elections

Thursday, 16 January:

0745-0800:	Coffee, Donuts, & Danish
0800-0945:	XWG Sessions (CAISWG meets with CAIS V2 Contractor)
0945-1000:	Break Coffee
1000-1130:	XWG Sessions (CAISWG meets with CAIS V2 Contractor)
1130-1300:	Lunch
1300-1400:	XWG Sessions (COMPWG meets with CAIS V2 Contractor)
1400-1500:	XWG Sessions (STONEWG meets with CAIS V2 Contractor)
1500-1515:	Break Coffee and Soft Drinks
1515-1600:	XWG Sessions (GACWG meets with CAIS V2 Contractor)
1600-1700:	Wrap-up

APPENDIX B
ATTENDEES
KIT/KITIA Meeting
14-16 January 1986

KIT Attendees:

BELZ, Frank	TRW
CHEVERS, Ed	NASA
EMERSON, Matt	NAC
FERGUSON, Jay	Department of Defense
FOIDL, Jack	TRW
HART, Hal	TRW
HOUSE, Ron	NOSC
KRAMER, John (Jack)	IDA
KRUTAR, Rudy	NRL
MORSE, Dit	Oracle Corporation
MUMM, Hans	NOSC
MUNCK, Bob	MITRE
MYERS, Gil	NOSC
MYERS, Philip	AJPO
OBERNDORF, Tricia	NOSC
POGGE, Dave	NWC
PRITCHETT, Gary	SofTech
ROWE, Kenneth	Department of Defense
STILES, Lloyd	FCDSSA-SD
SZYMANSKI, Raymond	AFWAL/AAAF-2
TARDY, Jean	National Defense Hq. Canada
TAYLOR, Guy	FCDSSA-DN
THALL, Rich	SofTech

WOOD, Bill	Software Engineering Institute

KITIA Attendees:

ABRAMS, Bernard	Grumman Aerospace Corp.
BAKER, Nick	McDonnell Douglas
DRAKE, Dick	IBM
FAINTER, Bob	Virginia Polytech
FISCHER, Herman	Litton Data Sytsems
FREEDMAN, Roy	Hazeltine Corp.
GALLO, Ferdinando	Bull France
GARGARO, Anthony	CSC
HARNEY, Terry	Hughes Aircraft
HARRISON, Tim	Texas Instruments
HORTON, Michael	System Development Corp.
HUSETH, Steve	Honeywell/SRC
JOHNSON, Ron	Boeing Company
LINDQUIST, Tim	Arizona State University
LYONS, Tim	Software Sciences Ltd.
McGONAGLE, Dave	GE
NURMI, Esa	Oy Sofplan AB Finland
PLOEDEREDER, Erhard	IABG West Germany
REEDY, Ann	PRC
ROUBINE, Olivier	Informatique Internationale France
RUDMIK, Andy	GTE
RUDOLPH, Bruce	Norden Systems
SIBLEY, Edgar	AOG Systems

STEIN, Larry	Aerospace Corp.
WILLMAN, Herb	Raytheon Company
WINNICK, Charlotte	Norden Systems
YELOWITZ, Larry	Ford Aerospace

VISITORS IN ATTENDANCE

ALDEN, Tony	TRW
ANDERSON, Les	NOSC
ANDREWS, Dave	Andyne Computing Limited Canada
CHLUDZINSKI, John	Institute for Defense Analyses (IDA)
CLOSE, Brian	British Defence Staff
CLOUSE, Jeff	IDA
COLLOM, David	NOSC
KERNER, Judy	TRW
LAKE, Mike	IDA
LEE, Marshall	Gunter Air Force Base
LEGRAND, Sue	SofTech
ROBINSON, Max	IDA
ROSKI, Steve	Logicon
SCHAAR, Brian	Techplan
STEVENSON, Bob	Gould
TADMAN, Frank	TRW

APPENDIX C
MEETING HANDOUTS
14-16 January 1986

1. NASA Space Station Software Requirements, Edward Chevers, Johnson Space Center, January 1986.

2. DoD Requirements and Design Criteria for the Common APSE Interface Set, KIT/KITIA, 13 September 1985.

3. SVID As A Basis For CAIS Implementation, H. Fischer, December 14, 1985.

4. PCTE Conformance to the RAC, Members of the Environment WG, Ada-Europe, 8 January 1986.

5. KIT Membership Address List dtd. 9 January 1986.

6. KITIA Membership Address List dtd. 9 January 1986.

7. KIT/KITIA MINUTES, Meeting of 10-12 September 1985, Saratoga Springs, New York.

8. Draft KIT/KITIA Glossary, updated 11/29/85.

9. Ada Run Time Support Environments and a Common APSE Interface Set, Charles McKay and Rodney Brown, University of Houston, Working Papers.

10. A Study to Identify Tools Needed to Extend the Minimal Toolset of the Ada Programming Support Environment (MAPSE) to Support the Life Cycle of Large, Complex, Distributed Systems Such as the Space Station Program, Charles McKay, University of Houston, Interim Progress Report.

11. E&V Status Report, December 1985.

12. CAIS Version 2 Contract Status Report, SofTech, 14 January 1986.

13. A Basis for a Portable Common Tool Environment (PCTE) Design Guidelines, PCTE Project Team, and Overview of PCTE: A Basis for a Portable Common Tool Environment, PCTE Project Team, ESPRIT Technical Week 1985.

14. RAC Rationale to Section 4 Entity Management Support, Section 5 Program Execution Facilities, Section 6 Input/Output, not dated.

15. Ada Interoperability Survey, GACWG, 15 January 1986, Draft.

KITIA MEETINGS
SAN DIEGO
JAN, 1986
EXECUTIVE COMMITTEE MINUTES

An executive committee meeting was called to order by H. Fischer. Numbered working group representatives present, forming the executive committee were:

1. B. Abrams
2. D. McGonagle
3. R. Johnson
4. N. Baker

The meeting began with only the chairman and the four numbered working group representatives. P. Myers was subsequently invited to attend after applicants and agenda issues were initially discussed. The group considered applications from the following applicants:

N. Gallo	Unanimously accepted.
T. Harrison	Unanimously accepted.
T. Lindquist	Unanimously accepted.
L. Stein	This gentleman and S. Glaseman proposed to exchange their alternate KITIA and primary KITIA representative roles (Aerospace Corp.), respectively. This was accepted.

If the acceptance of all applicants would cause a seat problem, it was suggested that a review of attendance and contribution be made. It was later suggested that since T. Lindquist was funded to attend by the Government, he could be possibly considered a KIT (vs KITIA) attendee. This suggestion defused the worry about seat count.

There was discussion of a general concern of lack of KITIA activity in the recent months. There was discussion of what to do about the SofTech award (NOSC contract). Some uninvolved industry members were concerned that politics had weighed too heavily in the government's selection. It was noted that KITIA could actively and constructively seek to help guide and influence the SofTech contract, since P. Oberndorf is also the contract technical honcho. This was felt to be positive. (Politically, it was noted, if CAIS conversion of ALS were not done under NOSC control, but instead by Army or other Navy sponsorship, KITIA influence would then be nil.)

A recommendation was made to change the numbered working groups. (See KITIA minutes following.) Realignment was to be by RAC working sections. Flexibility in group allegiance was recommended. A proposal was made to provide a KITIA executive board consisting of the Chair and four independently elected directors, instead of the numbered group chairs. Each numbered task area (or its replacement) would then have both KIT and KITIA leaders, and a joint spokesperson.

A proposal was made to continue the primarily joint KIT/KITIA meetings. This eventually was decided to include nominally one hour of KITIA numbered group time to make a "statement of industry/academia differences/support views for the government (focused on current CAIS 1 and CAIS 2 issues)." A subsequent general (1 hour nominal) KITIA meeting was proposed to review the functional area differences and areas of support, to provide joint input to the sponsor.

A discussion occured on the lack of representation on KITIA by a few of the major academia institutions shaping the environments field (notable examples included CMU, Stanford, Berkeley, Columbia, and NYU.) This discussion was repeated for P. Myers, so the AJPO could be aware of the concern. A recommendation was made to consider inviting temporary members for specific support in task areas, such as to contribute to the security work. The suggested KITIA topic for the Atlanta meeting was to have a session on Secure OSes, and how they apply to CAIS 1 and 2 solutions with respect to RAC requirement 2.8.

H. Fischer was proposed by the executive committee as their nomination for KITIA chair election.

The executive committee adjourned.

KITIA MEETING MINUTES

The KITIA meeting was called to order by the chairperson, who first reviewed the activities of the executive committee meeting. A suggestion was made to get a clean and complete set of attendance records (presumably for better understanding of which representatives are participating regularly; in response to the concern of KITIA seat usage.) This task was to be passed onto TRW as a support contractor activity.

The question was raised 'What is the KITIA's current goal?' We should articulate our goal. A member suggested that it was 'to provide input guidance [to DoD or to KIT -hf] to work under contract [e.g., CAIS 2 -hf]'. Another suggested we should take documents for review at each meeting and split up to review them.

Discussion went back to the executive meeting proposal to realign numbered working groups according to the RAC. It was proposed that the original group definitions be retained, but with the ability of people to move about. A chart was drawn up to show numbered working group definitions and RAC alignment. Its final form, after open discussion is as follows:

WG 1	Process Management	(RAC sect. 5)
WG 2	Entity Management	(RAC sect. 4)
WG 3	I/O (& user interface)	(RAC sect. 6)

WG 4 MISC (Security, distribution, knowl base) (RAC sects. 1, 2, & 3)

All members of KITIA are to decide whether they want to stay in their current WG n (nth working group) or to change. Specification is to be made before or at the Atlanta meeting. [Requests to change working group are to be made by net to: (a) H. Fischer, (b) Hans Mumm (to track changes), (c) old WG chair, and (d) new WG chair.

A motion to adopt the new numbered working groups and policy as above passed. Discussion next focused on the executive committee suggestion for a board of directors not aligned with numbered working groups. With the above structure, open discussion went away from supporting the independent executive committee idea, back to the concept of numbered working group leaders forming the executive committee. A motion was made to leave the executive committee as is: composed of numbered working group chairs; it passed.

Working group elections were discussed next. Agreement was reached to suspend numbered group elections until the Atlanta meeting, because of the accepted proposal to change to the above policy on working groups.

Chairperson election occured next. H. Fischer was nominated and elected.

Discussion next covered the support contractor support. There was complaint about the lack of minutes from preceeding meetings, and the lack of specific support contractor coverage of KITIA-only meetings. Concern was expressed that the meeting recorders and minutes writers should be knowledgeable about the subject of the discussion areas, so that the coverage in the minutes be of the important and essential items. [Agreement was obtained from the sponsor and support contractor to take reasonable action to alleviate the shortcomings. -- HF] A request was made for the support and CAIS 2 contractors to provide regular progress reports early in each meeting. Concern was made of whether there would be minutes from the CAIS 2 contractor status report earlier in the week.

Concern was expressed that the CAIS 2 contractor discussions, earlier in the week, seemed to be exclusively between R. Thall and the audience, rather than between the SofTech parties responsible for the individual work areas, and the audience. [I recall that I was asked to provide this concern, and the preceding concerns, to Tricia, Hans, and Phil: I am doing so via these minutes -- HF] H. Fischer agreed to provide his notes as substitute minutes for the meeting. [Any omissions or inaccuracies therein are accidental or due to my illegible handwriting -- HF]

MINUTES OF MEETING

COMPWG

KIT/KITIA

JANUARY 13-17, 1986

ATTENDANCE

Bernie Abrams - Chairman
Bob Fainter
Jack Foidl
Delwin McCracken
Lloyd Stiles
Ray Szymanski
Guy Taylor
Larry Yelowitz

Jack Foidl and Guy Taylor are working on the traceability matrix between CAIS I and the RAC (Requirements and Criteria). A modified tool called RAVE will be used with a database. The database is DBASE2 or R5000. The tool will provide paragraph to paragraph traceability.

A presentation by Bernard Abrams on Testing Principles related to CAIS was reviewed by the group. The presentation will be given to KIT/KITIA in the April meeting. The presentation reviewed the various criteria for selecting test cases including both black box (input based) and white box (code based) methods. The applicability of various methods to testing CAIS implementations was discussed.

A net message will be sent by B. Abrams to the COMPWG to test the address file. Anyone not receiving the messge in the first week after the KIT/KITIA meeting should notify ABRAMS and TRWKIT.

Lloyd Stiles presented a draft paper on Quality Assurance Guidelines. The paper included a list of Software Quality Factors.

COMPWG met with Softech to review related issues in the CAIS contract. Softech has no requirement to formally test the prototypes. They will build a "Flogger", a tool to test every interface. COMPWG will continue looking into test issues because production CAIS implementations will have to be tested. The E & V group has the task of testing CAIS. COMPWG has an advisory role.

John Kelly of Softech in Waltham is working on formal semantics. The method of specifying the formal semantics has not yet been decided.

The things being worked on for the next quarter are:

Testing Methods	B. Abrams & R. Fainter
Traceability Matrix (RAC to CAIS)	J. Foidl & G. Taylor
Quality Assurance Guidelines	L. Stiles
Testability of RAC	R. Drake

Prepared By: B. Abrams

KIT/KITIA MINUTES
MEETING OF 15-17 APRIL 1986
ATLANTA, GEORGIA

AGENDA: ƎEE APPENDIX A

ATTENDEES: SEE APPENDIX B

MEETING HANDOUTS: SEE APPENDIX C

TUESDAY, 15 APRIL 1986

1. OPENING REMARKS

- o Tricia Oberndorf, KIT Chairperson, brought the meeting to order.
- o New KIT/KITIA members and visitors were introduced. Tom Smith represented the MITRE Prototyping Team for the KIT and Don Vines was present for Honeywell's representation to the KITIA. LCDR Dave Endicott of PMS-408 and Steve Atkins of CDC were present to support the ALS/N presentation.

2. KITIA CHAIRMAN REPORT

Herm Fischer, KITIA chairperson reviewed the status of KITIA activities:

- o The issue of the KITIA renewal is important to some of the KITIA membership. Some of the members are concerned ovcr their legal status.
- o The KITIA members that attended the Gould demonstration of their CAIS prototype found it very interesting.
- o Herm noted there has been a lack of interest in security related issues on the part of the KITIA. He will be presenting a report on the implications on UNIX [1] to meet the requirements of the Orange Book (DoD Trusted Computer System Evaluation Criteria).

[1] UNIX is a registered trademark of AT&T Bell Laboratories.

[2] Ada is registered trademark of the U.S. Department of

Defense.

3. EVALUATION AND VALIDATION TEAM REPORT

Ray Szymanski, E&V Chairperson, presented a status report of the E&V activities:

- A new version of the E&V Status Report is not available at this time.
- The E&V Team made a formal presentation and supported a Birds-of-a-Feather meeting at tne SIGAda in Los Angeles. The next formal presentation is scheduled for the Ada [2] Europe conference in Edinburugh, Scotland.
- Two additional Air Force agencies are expected to join the E&V Team in the near future.
- Expect to have two contracts awarded by the end of the fiscal year. the second "Sources Sought" synopsis on the CAIS Validation procurement is now out; expect a may RFP release for this work. The second "Sources Sought" for the Ada Compiler Validation Capability should appear in the CBD at any time now.

4. CAIS VERSION 2 DESIGN (SOFTECH) REPORT)

- SofTech has completed their analysis of the Requirements and Design Criteria for the Common APSE Interface Set document and will presenting their findings later in this meeting.
- A preliminary design of a CAIS prototype has been submitted to NOSC.
- SofTech is continuing to collect data for their Issues Reports which will be presented at the July meeting.

5. KIT SUPPORT CONTRACTOR (TRW) REPORT

- TRW has completed their initial traceability analysis of the RAC and CAIS Version 1 which will be the subject of a later presentation.
- Frank Belz is supporting the response formulation to the comments submitted on CAIS Version 1 Public Review cycle.
- TRW will present a status report on their prototyping activities during tomorrow's session.

GRAMBO STRIKES !!

Tricia Oberndorf reported that as a result of the Grann-Rudmann amendment there has been a reduction in her planned budget resulting in reductions in the scope of both the SofTech and TRW prototyping activities. Neither group will receive enough resources for a full implementation. MITRE has submitted a proposal for a PCTE implementation on a bare machine which also cannot be funded.

6. WORKING GROUP REPORTS

- CAISWG - Jack Kramer reported the CAISWG has been working on a draft Rationale document that incorporates previous work of Tim Harrison and Erhard Ploedereder. a draft CAIS Reader's guide has been circulated for review but still needs improvement. The CAISWG during the next quarter will be continuing the Rationale development and supporting the Standardization process including formulating responses to the comments received during the Public Review period.

- RACWG - Hal Hart reported that the 13 September 1985 version of the Requirements and Criteria document is the latest version of the RAC. There have been very few change Proposals submitted to date. a draft integrated RAC Rationale document will be circulated to the RACWG for review. An updated version is expected to be available at the end of the month. The RAC-COMMENT account on the MilNet had some comments for RACWG review but most were also received from Ada Europe for RACWG review.

- GACWG - Matt Emerson reported the GACWG still requests input via the Ada Interoperability Survey forms which will be available again at this meeting. The Transportability Guide is now being transferred to the KIT Support Contractor for finalization. A marked-up copy will be distributed to the GACWG for review during this meeting. The final document is expected to be available in July for KIT/KITIA review.

- STONEWG - Ann Reedy reported the STONEWG is experiencing difficulty in obtaining a critical mass to support interim meetings so they can generate an integrated document rather make progress on independent sections.

- COMPWG - Bernie Abrams gave a status report of the COMPWG activities since the last KIT/KITIA meeting. They have completed the initial RAC/CAIS traceability analysis which will be presented later. Bernie will also give a presentation on test methods for CAIS compliance. The paper on formal semantics for the CAIS has been completed by Drs. Roy Freedman, Tim Lindquist, and Larry Yelowitz.

- DEFWG - Hans Mumm reported that the DEFWG is expecting updates to the Glossary from the RAC and Transportability revisions. The CAIS Version 2 contractor may also have inputs fcr consideration.

7. GENERAL ANNOUNCEMENTS

- Some future meetings that may be of interest to KIT/KITIA members include the Ada Europe meeting at Edinburugh, Scotland 6-8 May and the NASA Space Station conference 2-5 June in Houston, Texas. The ASEET Symposium 10-12 June is a good forum for academics to review with current status of Ada education and training. Additional future activities to keep in mind include the SIGAda meeting 23-25 July in Pittsburgh, Pennsylvania and the SIGAda sponsored Future APSE Workshop in Saratoga Springs, New York, 9-12 September.

- The meeting schedule remains as follows:

1986	July 7-10	San Francisco
	September 22-25	Minneapolis
1987	January 17-20	San Diego

- Dianna Peet, as the Control Data Corporation local host, welcomed the KIT/KITIA to Atlanta. a demonstration of the Gould CAIS prototype is planned for Wednesday and Thursday evenings for those that could not attend the presentation in Fort Lauderdale.

- New organizations joining the KIT include representatives form NASA Johnson Space Center and MITRE Corporation.

- The MILNET address problems are under repair. The Other Interested Parties (OIP) mailing list is available for use as well as the CAISV2 list.

- Regarding the CAIS Standardization process there have been 312 formal comments submitted regarding CAIS Version 1 with an additional 229 informal comments. Comments may be viewed through the <KIT-INFORMATION> directory in the files FORMAL-CAISV1-COMMENTS. The CAISWG is continuing to formulate responses to the received comments.

8. AJPO COMMENTS

LCDR Philip Myers, Navy Deputy to the Ada Joint Program Office, addressed a number of issues in the following areas.

- The CAISWG members receive additional inputs other than the formal comments received. Will they be considered for inclusion in the CAIS^ since the KIT/KITIA is in a production mode for delivering a final document for submission in December 1987, changes to correct small problems should be considered. Major fundamental changes would be a real problem. The CAISWG will review the technical comments submitted. Those comments of a "political" (i.e., policy) nature will be reviewed by the SWG. Therefore, all submitted comments will be reviewed.

- The issue of security and its impact on the CAIS is still a thorny issue. Coordination with Trusted Computing Base (TCB) developers will be initiated to identify CAIS related issues.

- The COMPWG needs to maintain a close coordination with the Evaluation and Validation Team to insure there is no unnecessary duplication of effort.

- Regarding the MilNet support, the AJPO is working the funding issues. KIT/KITIA personnel are reminded not to directly address requests to the <DIR-REQUESTS> personnel but to work with the KIT/KITIA chairs for prior approval before submitting for AJPO approval. Requests without prior KIT/KITIA chair approval will be rejected by the AJPO.

- The AJPO is continuing to monitor the potential impact of the ITARs on the work of the KIT/KITIA. The STARS Taxonomy is listed as an ITAR product and the use of derived data form ITART products may cause problems.

- There is now a Statement of Intent among the Allies (U.S., U.K., Canada, Spain, West Germany, France, Greece and Italy) to do cooperative work on Ada Environments. The next step will be to develop a Memorandum of Understanding to establish the specific scope and responsibilities of the signataries.

- Although the AJPO views the CAIS/KIT/KITIA as the flagship activity of the program, funding constraints are impacting the scope of what can be accomplished. AJPO is trying hard to identify alternative sources of funding. With the awarding of the E&V Team contracts, the contracting base will have increased by 100%.

- Regarding the KITIA membership, although the KITIA wa originally organized on a competitive selection basis the question if it continues on a sole-source basis is allowable. At issue is the KITIA considered as being a de facto Federal Advisory Committee (which has specific approvals required for its existence). The Ada Joint

Program Office may not be able to resolve the situation. AJPO needs feedback from the companies regarding the impact of the KITIA legal/quasi-legal status and the impact of two additional years support.

- Within the context of available resources and product focus, the STONEMAN II activities may be terminated as not directly related to the team product.

- COL Joseph Greene is the new Director of STARS reporting from previous duty as the Deputy Director at the DoD Computer Security Center.

9. MORNING BREAK

10. ALS/N PRESENTATION

Doug Wrege introduced Steve Atkins, ALS/N Program Manager at CDC, who presented an overview of their ALS/N activities.

- ALS/N is originally planned for implementation on a VAX host with the AN/UYK-43, 44, and 14 as the target machines. Other host/target configurations are possible at a later time. ALS/N is based on the 500,000 LOC of the Army ALS. ALS/N will incorporate Ada/L for 32-bit machines, Ada/M for 16-bit machine, an expanded tool set and a distribution concept which will be explained in greater detail. TRW as a subcontractor to CDC is developing a text editor, text formatter, report generator and later, additional configuration management tools. The run-time support area is one in which CDC is applying a lot of effort. ALS/N takes the Army ALS as a baseline so the ALS KAPSE is the basis of the environment.

- The program is structured for a Build 1 and Build 2 development. Build 1 is targeted for a single mainframe architecture while Build 2 is targeted as a distributed environment. Some of the problems being addressed in Build 1, such as memory reach, are contribution to a better understanding of the Build 2 issues.

- ALS/N is segmented into the Minimal Ada Programming Support Environment (MAPSE) and Run Time Environment. The MAPSE is composed of the Language Processor, Text Manipulation, Separate Compilation Support, User Access Support, Code Manipulation, MTASS Interface, and the MAPSE Run Time Environment. The ALS/N Run Time Environment is composed of the Run Time Operating System (RTOS) and the Run Time Application Support (RTAS). The RTAS supports the debuggers, loaders, overlay management and performance measurement types of tools. CDC is extending the language processor area and the RTOS.

NOTE

Doug Wrege observed that the following data has been presented to the Navy (PMS-408) but has not been forr.ally approved and therefore does not represent an official Navy position on the technical composition of the ALS/N.

- o In multi-programming, CDC plans to have a single-tiered scheduler in which the multiple programs appear to the scheduler as a single program. The point is to slide the priorities of the program tasks between the programs. The relative priority is scheduled dynamically. Therefore, the scheduler really doesn't understand programs, only tasks. The priorities are dynamically adjusted at run time to make up the system parities. Multiple tasks with the same priority have not as yet been addressed since the design is not yet complete. That situation may become a FIFO process.

- o Ada tasks need visibility into other Ada tasks for data objects, other library units, or other compilation units. Distribution on task boundaries is not reasonable since this would require an extremely complicated run time executive to satisfy Ada semantics or it would require significant constraints on the use of the Ada language.

- o Some of the distribution problems include how to call a procedure of a library unit that is in another machine. Another is ho to handle dependent tasks. Visibility rules are not amenable to partitioning and intra-machine rendezvous can be extremely complex.

- o A key to the CDC approach is that there is no knowledge on one machine of the detailed state of any other machine. Some of these ideas are based on the work at Honeywell. The unit of distribution is the library unit which simplifies what is visible to other units. All cross machine access is to objects identified in a package specification. It also simplifies tasking as the task object is always located with masters. This simplifies termination rules and entry queues can be localized on specific machines.

- o Other characteristics of this approach are it results in a static configuration, the distribution is specified at compile time and is transparent to a program.

- o One of the single processor (Build 1) problems that relates to distribution is the AN/UYK-43 cross-phase problem. The AN/UYK-43 has a 512K address space. How do you map a program which is bigger than one address space (phase) and still call two different address spaces (cross-phase). Solving cross-phase procedure calls is similar to distributed problems where you have two tasks running in different machines. The CDC ALS/N team is really interested in addressing these RTE issues.

- LCDR Philip Myers pointed out that although this problem is specific to the 43 machine architecture, the military specification process is so long to effect changes the issues must be addressed. The ALS/N project has developed a solution that is not being addressed by any other group except the ARTEWG.

- Doug explained the approach to making a procedure call across phases is through a surrogate routine (or routines) which gets access to system maps to transfer control from procedure A to get to the other phase and c all the procedure in package B and return results to package A. This provides flow of control and parameter passing across phases and can be applied to distributed processing. the RTE exec is not really distributed but rather can be viewed as a local executive with a fancy communication system.

- In summary, CDC believes this is a low risk approach since they do not really distribute the operating system. They expect improved portability, maintainability and survivability.

11. LUNCH BREAK

12. SOFTECH PRESENTATION

Gary Pritchett started the SofTech report with the status of their CAIS prototype effort and progress on the issue reports to be presented at the next meeting.

- Due to budget constraints imposed by the Grann-Rudmann restrictions, the CAIS prototyping activities have been redirected.

- The previous goal of a functionally complete CAIS Version 1 prototype has been deleted from the present goals. The prototype now is intended to provide prototype support for the CAIS Version 2 designers. Implementability of CAIS Version 1 has already been demonstrated by the Gould and MITRE products. The valuable resources are being directed to the CAIS Version 2 effort. The evolutionary approach from Version 1 to Version 2 remains in effect but there is now no deliverable for a functionally complete Version 1 prototype. Activities this year will focus on constructing a Version 1 prototype including the node model and process model as a basis for Vion 2 prototype next year. It is not clear at this time how the I&T goal will be achieved but it remains a goal for the project. A good way to demonstrate I&T is to move tools around but there are no tools to move.

- The CAIS Version 2 prototype is to be functionally complete including security, distribution, node typing and demonstrate I&T. The purpose is to demonstrate CAIS Version 2 implementability and provide a basis for tools studies. The initial performance is not a high priority. Following completion of the initial product there will be a tuning of the prototype to realize near production level performance and provide a model for other implementations. The Version 2 product can provide a basis for tool development and use.

- The SofTech contract is composed of a series of Delivery Orders which are written to accomplish a specific task. Delivery Order 1 called for a draft design of a CAIS Version 1 prototype which has been submitted to NOSC. Detailed design in the Node and Process models will continue. There are some mechanisms in the ALS KAPSE that would support a prototype and it is still under evaluation. SofTech has not decided if they will take some ALS KAPSE code and add new code or modify the ALS KAPSE code. The prototype will use some of the design of the new ALS KAPSE and most probably have some of the code as well.

- The prototype will be the basis for the Version 2 prototype and provide a base for design experiments such as performance instrumentation and design validation to verify multi-user architecture and the distribution architecture. This prototype will evolve to CAIS Version 2 when the draft is available.

- The only schedule change is the removal of the CAIS Version 1 prototype deliverable. The Node and Process model prototype is scheduled for delivery December 1986 and the CAIS Version 2 prototype delivered December 1987.

- Delivery Order 3 is for the construction of Issue Reports related to the design of CAIS Version 2. Issue Reports will be generated for node typing, distribution taxonomy, logical device drivers, performance tradeoffs, access control, security and I/O priorities.

- Rich Thall presented the results of the SofTech Requirements and Criteria (RAC) document. Their goals were to analyze conformance of the RAC to CAIS Version

1 and to interpret and comment on the RAC contents. The RAC wad divided into functional areas and ranked on a scale from 0-10 where 0 equals no conformance and 10 equals full conformance.

- The general design received highest marks for the uniformity of the CAIS and its extensibility. Lowest grades were assigned to the security and technology compatibility.

- In the syntax and semantics area the CAIS scored high in syntax but lower in semantics due to incomplete responses, exceptions, and enumeration of pragmatics.

- In entity management higher marks were received for operations and entities and relationships with very low points for typing, transactions, histories and robustness.

- In program execution the lowest points assigned were for monitoring (receiving a 6) with higher grades for termination, communication, synchronization and program activation.

- In input/output block devices scored well but low marks were assigned (lower than 4) to data path control, unit devices, unit transmission and lowest (0-1 level) was assigned to block transmission, entity transfer, and general I/O.

- Rich Thall presented the argument that the RAC has multiple and conflicting personalities for the design of the CAIS Version 2. This is base on the requirements for piggy-back as well as stand alone implementations, the features required by the RAC along with performance and security requirements, and the flexibility versus Interoperability and Transportability. He believes this is a reflection of the varying user needs that were considered in the generation of the RAC. He, therefore, proposed that one CAIS subset be designed-in during the Version 2 design process so that a user could delete features he doesn't want or need. If a user doesn't need security mechanisms he could delete them form the CAIS implementation. He expects there would still be interoperability with full CAIS hosts yet possibly more appropriate for workstation applications. This could occur at implementation time, installation, CAIS startup, session initiation or process startup.

- Another proposal for consideration was the definition of additional interfaces for Logical Device Drivers that would be written in Ada, be installed on-the-fly and be portable between CAIS's (but not devices).

13. NAMED WORKING GROUP MEETINGS

14. ADJOURN FOR DAY

WEDNESDAY 16 APRIL 1986

15. TRW REPORT

Hal Hart gave an overview of the TRW KIT/KITIA support contract including tasking for CAIS prototyping, the RAC, CAIS maintenance, the Transportability Guide completion, MILNET and other working group support.

Frank Tadman continued with additional details on the TRW prototyping activities including a discussion of the following areas.

o The plan is to implement a basic CAIS. Almost complete are node management, structural nodes, list utilities and supporting lower level packages. Under development are attributes and text I/O.

o Limitations on this prototype are there is no access control enforcement, no exclusive access enforcement, no time limits or iterations and all structures are built in main memory.

o The prototype is built on SUN workstations on a division of Berkely 4.2 UNIX with code generation from the Verdix Ada Development System (VADS).

Some of the risks associated with CAIS acceptance were described as:

o Performance - since the CAIS is more complex than typical operating systems the standard techniques for achieving efficient operation may not be sufficient and new algorithms and/or architectures may be required. This efficiency can be especially important in piggyback (layered) implementations.

o Security - the CAIS security mechanisms must be adequate to support the DoD security policies. They must also be implementable with acceptable performance and must be usable by both the tool writer and the APSE user.

o Appropriateness - the CAIS must not omit important interfaces nor provide inappropriate ones an must be usable by the tool writer (aside from the security aspects). The underlying model may not admit clear and useful presentations to the users since the CAIS network is more difficult to visualize than a hierarchial filesystem. Also, the access control mechanism may be difficult to use.

Frank then provided a detailed discussion of the ranges of design objectives including prototype functionality, portability of CAIS implementation, prototype instrumentation, security, performance, administrative tool support, capacity, host operating system support, tool construction and re-hosting, and CAIS Version 2 anticipation. The status of the TRW prototype across these design ranges was then presented to display the design goal and current implementation status. Frank then described TRW's approach to the previously identified risk areas:

- Performance - investigate alternative algorithms for performance critical areas such as access control, caching, and process control and to simulate the effect of various prototype architectures on performance.

- Portability - develop several inner portability layers to support the portability of various portions of the CAIS implementation. Also under consideration is the re-hosting of the prototype to at least one other host/operating system.

- Security - investigating various development options including:

  - building the CAIS on an "existing" TCB

  - building the CAIS on a modified TCB

  - building the CAIS on a new TCB

  - building the CAIS as a new TCB

- Appropriateness - port existing tools to the CAIS and build tools on the CAIS

16. RE-CONVENE INTO NAME WORKING GROUPS

17. LUNCH BREAK

18. RE-CONVENE INTO NAME WORKING GROUPS

19. AFTERNOON BREAK

20. RE-CONVENE INTO NAMED WORKING GROUPS

21. ADJOURN

THURSDAY, 17 APRIL 1986

22. COMPWG PRESENTATION

Two presentations were made by the COMPWG for the methodologies and the initial traceability analysis.

- o Bernie Abrahms presented an overview of the objectives and methodologies of various testing approaches applied to software. Historically, software testing can be a significant part of the software cost and is most critical for real time embedded systems. The tradeoff that must be made is how much testing can be afforded while confident he the testing level insures objectives are met. The differences between "black box" (input driven) and "white box" (code driven) testing was explained. Within the large scope of the CAIS a strategy may be to test every function at least one time. The testing may be "black box" consisting of about 5 tests per function (450 functions resulting in about 2250 tests.

- o DeWayne McCracken of TRW presented the results of a preliminary RAC/CAIS Version 1 traceability analysis. This activity was performed as part of a COMPWG plan to formulate a baseline data base between the CAIS and the RAC that could be updated during the evolution of CAIS Version 2 to insure the completeness of the Version 2 specification. The database could be automated to include candidate test sets linked to the requirement and its corresponding CAIS fulfillment. The preliminary results showed less then 40% of the RAC requirements were met [subsequent resolution of RAC/CAIS interpretation issues and establishment of an equitable way to statistically compile results categorized as completely versus partially met raises this figure to approximately 63% with approximately 30% attributed to the formally "deferred" topics to be included in Version 2 leaving approximately 7% of the RAC requirements not addressed]. An additional issue raised in this presentation included the lack of a weighing criteria for the RAC requirements (in this study all RAC requirements retained the same value).

23. USER INTERFACING AND WINDOWING PRESENTATION

Herm Fischer presented an overview of the current approaches to user interfaces and windowing. The evolving "standards" that are emerging can impact tool development. Herm's selection for comparison included the MacIntosh, X, PCTE Microsoft, and ADREW implementations. The MacIntosh interfaces were described as low level and proprietary. Herm produced copies of the PCTE and ANDREW interfaces to demonstrate the relative size of the required interfaces to support this functionality. ANDREW has 71 tool interfaces while X has 142 and PCTE has 144. Each has a different form of input and internal process. PCTE has the ability to store bit-mapped graphics.

24. MORNING BREAK

25. INFORMAL REPORTS

- Dick Drake reported some preliminary results of using a CAIS environment with some tools. The user was satisfied with the environment and the node model. The implementation was considered very fast. A developed backup/restore utility required about 1 minute for completion. Some feedback (objective evaluation of the user) or use of this environment was that the CAIS was no more difficult to understand then a typical operating system. "The environment was fast and reliable". Security was not yet implemented. A command line interpreter was developed with the environment which turned out to be very useful for testing. Some simple tools such as a line counter and prettyprinter were migrated to the CAIS environment. The feeling was that after one or two tools are converted it becomes relatively easy to migrate additional tool since the changes are about the same. The user was happy with the environment and interfaces.

- Rich Thall presented an update on the recently released ALS Release 3 system. The new database indexing method reduces the I/O access (disk seeks) by 50%. They have developed a frame file index. They have eliminated the database index server (ACP) in favor of locks which they believe is a more portable design. At the present time they cannot support multiple CPU use on a VAX cluster but this will be changing shortly. The ALS database will be able to be shared among multiple CPUs on a cluster. There are a number of tool improvements including compiler speedups which result in an overall improvement of two times over Release 2. The detailed ranges go from 30% to 9 times. There is better throughput with multiple users. SofTech now uses the ALS for configuration management for their 500K lines of code. On a VAX 780u are still compute bound while on an 8600 you realize the improvements. Performance on the MicroVAX is comparable to the 780 but you don't get thrashing.

- Dave Pogge reported on the evolution of software systems in use at the Naval Weapons Center at China Lake from the construction of special purpose processors with machine code to use of larger systems for code development and downloading into the target processor. He identified needs for their applications for representation clauses for 8-bit registers, standard I/O subprograms in the CAIS, Ada object code which runs without any operating system and an Ada development environment similar to the Microprocessor Development System.

- Sue LeGrand reported that NASA Space Station will have all operational software written in Ada at all the NASA centers. The RFP for the Software Support Environment is expected out by June. The environment is not a physical facility but a set of methods, tools, dbms and standards. These will become the components in a Software Production Facility. The initial SOW required "CAIS compliance" but this specific statement does not now appear although CAIS remains a strong consideration. The Ada Beta Team charter is being changed to provide a software engineering research capability for the Johnson Space Center (JSC). JSC is sponsoring a June conference to address technical issues related to Ada.

- Herm Fischer has available a paper that addresses UNIX as an architecture rather than a product for security evaluations based on a UniForum conference. Some presentations at the meeting included what would have to be done to UNIX to make it secure and how this could be done while still retaining compatibility with unsecured versions. There are only two approved operating systems with CDC and DEC in the hopper for approval. The paper has additional details based on Herm's notes on UNIX modifications. It may be interesting to have the CAIS architecture evaluated. Progressing through the various security levels would require additional modification to UNIX. It appears to Herm the NCSC will evaluate more than just products.

26. BREAK FOR LUNCH

27. KIT/KITIA WRAPUP SESSION

- CAISWG - Jack Kramer reported the CAISWG will be continuing their responses to the submitted comments on CAIS Version 1. They will also be addressing issues raised as a result of the some of these comments.

- STONEWG - Ann Reedy reported the STONEWG is looking to compare existing environments to the CAIS Version 1 and the RAC and to see if they can evaluate differences or similarity. They expect to have possibly two presentations for the next meeting.

- RACWG - Hal Hart reported RACWG has been meeting with SofTech to discuss their recommendations and will continue this dialogue. SofTech will be using the RAC Comment form as a platform to submit suggestions. Hal will send a message on use of the RAC-COMMENT account.

- COMPWG - Bernie Abrams reported the COMPWG is focusing more on the practical considerations of testing and verifying the CAIS since the CAIS is moving into actual prototypes. COMPWG plans a presentation on automated test methods. Additional areas for consideration will be testability of the RAC requirements, exercising of the Gould prototype and looking at knowledge based systems for generating test cases.

- GACWG - Matt Emerson reported the GACWG is still looking for inputs to the Interoperability Survey. The Transportability Guide is being turned over to the support contractor for finalization and expects a July distribution to KIT/KITIA. They will be discussing the Interoperability Guide this afternoon. They expect to receive some Ada language usage papers for consideration. Additional work will be performed on the Interoperability Guide next quarter.

27. KITIA REPORT

Herm Fischer reported he had received a paper form Honeywell on security which will be made available for the Public Report.

The purpose of the separate meetings is to discuss issues raised by Industry and Academia members to be brought to the attention of the KIT and Ada Joint Program Office. There are also some Working Group election results:

Working Group 1	Bernie Abrahms	Process Mgmt
Working Group 2	Andy Rudmik	Database
Working Group 3	Herb Willman	I/O
Working Group 4	No members	Existence TBD (formerly distribution and security issues)

Herm then presented a statement of KITIA views.

- Steamrollering - The statement that some comments are put into a technical bucket and some into a "political" bucket. Tricia Oberndorf reminded the KIT/KITIA that political comments are to be addressed by the AJPO and technical comments by the CAISWG but be addressed by the AJPO and technical comments by the CAISWG but ALL comments are to be addressed. Herm summarized concerns over "non-political" topics submitted by EIA, NSIA, DMA, and ACEC. The KITIA position is that the AIS should be promulgated for prototyping and experimenting but not as a Military Standard. Tricia indicated the concerns summarized (specific comments) are not in the "political" category. Jack Kramer described that since the DoD got into this area originally to try to reduce Life Cycle Support costs and they must address issues now rather then wait until all interfaces are identified. Herm replied that there is strong concern that user interfaces and data management interfaces not yet addressed and the DoD is going ahead with definition of a Military Standard that can be invoked on contracts. There is a specific concern within the KITIA that CAIS version 1 become a MIL-STD. This is not a reflection on whether CAIS Version is good or poor. Tricia asked if a year delay for additional experimentation would be enough for KITIA support. Herm indicated that may not be enough time to provide a user/data management interface and they would probably find six more areas that are absent. The prime concern of the KITIA remains that all comments are treated fairly by the reviewers.

o Subsetting - There was split consensus on subsetting within the KITIA. It is not clear the need for subsetting has been demonstrated. Mandatory access and distribution may be candidates for subsetting but not other areas as recommended by SofTech. There is some differences among the members on just what should be candidates for subsetting. The KITIA feel SofTech needs to prepare strategy papers on data model selection and rationale and on the typing model and rationale. Tricia pointed out this is the purpose of the Issue Reports. Some of the KITIA feels the subsetting is a result of the shortage of funding. Tricia locates this as a function of time.

o Piggybacking versus Bare Implementations - The talk of bare machine implementations seems to scare industry. Gould and PCTE seem to be a demonstration of co-existence with operating systems (rather then one or the other). The KITIA is reacting to continuous references to bare machine implementations. Industry is concerned with continuing to exist with large mainframe systems and supporting operating systems.

28. RECONVENE INTO NAMED WORKING GROUPS

29. MEETING ADJOURNED

APPENDIX A
AGENDA FOR KIT/KITIA MEETING
14-17 APRIL 1986
ATLANTA, GEORGIA

Monday, 14 April:

Demonstration of the Gould CAIS prototype at Gould, Fort Lauderdale, FL.
All KIT/KITIA members are invited to attend the demonstration

Tuesday, 15 April:

0800-0830:	Arrive and Settle Coffee, Donuts, & Danish
0830-1030:	General Business and Announcements - Introductions - Status Reports (Contracts, etc.) - KITIA Chairman Report - E&V Report - DIANA Chairman Report - XWG Chairmen Reports - Announcements - Meeting Schedule - Local Arrangements - New Business
1030-1045:	Break
1045-1145:	ALS/N Presentation (Doug Wrege)
1145-1315:	Lunch
1315-1445:	SofTech Report
1445-1500:	Break
1500-1700:	Named Working Group Meetings

Wednesday, 16 April:

0800-0930:	TRW Report
0930-0945:	Break
0945-1145:	Named Working Group Meetings
1145-1315:	Lunch
1315-1500:	Named Working Group Meetings
1500-1700:	Separate KIT and KITIA Elections
1900-2100:	Gould Prototype Demonstrations (tentative)

Thursday, 17 April:

0800-0900:	Testing Methods for CAIS (Bernie Abrams)
0900-1000:	User Interfacing and Windowing (Herm Fischer)
1000-1015:	Break
1015-1145:	Informal Reports (Dave Pogge - China Lake Ada Considerations, Herm Fischer - Security, etc.) and numbered working groups if time permits (CAISWG meets the CAIS V2 Contractor)
1145-1315:	Lunch
1315-1415:	Wrapup
1415-1430:	Break
1430-1700:	Named Working Groups
1900-2100:	Gould Prototype Demonstration (tentative)

Friday, 18 April

Named Working Groups as needed

APPENDIX B
ATTENDEES
KIT/KITIA Meeting
15-16 April 1986

KIT Attendees:

BELZ, Frank	TRW
EMERSON, Matt	NAC
FOIDL, Jack	TRW
HART, Hal	TRW
HOUSE, Ron	NOSC
KOCH, Chuck	NADC
KRAMER, John (Jack)	IDA
MUMM, Hans	NOSC
MUNCH, Bob	MITRE
MYERS, Philip	AJPO
OBERNDORF, Tricia	NOSC
PEELE, Shirley	FCDSSA-DN
POGGE, Dave	NWC
PRITCHETT, Gary	SofTech
SMITH, Tom	MITRE
STILES, Lloyd	FCDSSA-SD
SZYMANSKI, Raymond	AFWAL/AAAF-2
TARDY, Jean	National Defense Hq. Canada
TAYLOR, Guy	FCDSSA-DN
THALL, Rich	SofTech
WOOD, Bill	Software Engineering Institute

KITIA Attendees:

ABRAMS, Bernard	Grumman Aerospace Corp.
BAKER, Nick	McDonnell Douglas
DRAKE, Dick	IBM
FISCHER, Herman	Litton Data Systems
FREEDMAN, Roy	Hazeltine Corp.
GALLO, Ferdinando	Bull, France
HARRISON, Tim	Texas Instruments
HORTON, Michael	System Development Corp.
LINDQUIST, Tim	Arizona State University
LYONS, Time	Software Sciences Ltd.
McGonagle, Dave	GE
PLOEDEREDER, Erhard	Tartan Labs
REEDY, Ann	PRC
ROUBINE, Oliver	Informatique Internationale, France
RUDMIK, Andy	GTE
RUDOLPH, Bruce	Norden Systems
STEIN, Larry	Aerospace Corp.
VINES, Don	Honeywell
WILLMAN, Herb	Raytheon Company
WINNICK, Charlotte	Norden Systems

VISITORS IN ATTENDANCE

ATKINS, Steve	CDC
ENDICOTT, Dave	PMS-408

APPENDIX C
MEETING HANDOUTS
14-16 January 1986

1. CAIS 2 Progress Report, SofTech.

2. Multi-programming and Distributed Ada, Control Data Corporation.

3. Draft, A Review of the 1986 UniForum Panel: a Secure UNIX System and Implications on CAIS and RAC, H. Fischer, February 9, 1986.

4. Notes on Testing Methods for CAIS, Bernard Abrams, COMPWG, April 4, 1986.

5. KIT Membership Address List dtd. 10 April 1986.

6. KITIA Membership Address List dtd. 10 April 1986.

7. KIT/KITIA MINUTES, Meeting of 14-16 January 1986, San Diego, CA.

8. Review of NASA Ada Status for SIGAda, S.A. Gorman, February 26, 1986.

9. Rationale for the DoD Requirements and Design Criteria for the Common APSE Interface Set (CAIS), Draft, KIT/KITIA, 13 September 1985.

MINUTES OF MEETING

COMPWG

KIT/KITIA

APRIL 14-17, 1986

ATLANTA, GA

ATTENDANCE:

B. Abrams - Chairman
R. Drake
R. Freedman
J. Foidl
D. McCraken
R. Styles
R. Szymansky
G. Taylor

Highlights of Activities in Previous Quarter

The traceability matrix (CAIS features to RAC requirements) was completed by J. Foidl, D. McCraken, and G. Taylor.

The paper on Applying Semantic Description Techniques to the CAIS by Abrams, Freedman, Lindquist, and Yelowitz was completed. It will be presented at an IDA conference and will be sent to the IEEE transactions on Software Engineering.

Presentations

COMPWG gave a 1 hour presentation to KIT/KITIA. B. Abrams spoke on testing methods related to CAIS. D. McCraken spoke on the results of the Traceability Matrix.

Presentations for Next Time

A COMPWG presentation is planned for the July meeting. L. Styles will speak on Quality Assurance of CAIS. R. Drake will speak on Automated Techniques for Testing.

Activities for Next Quarter

L. Styles and R. Drake will work on preparation for the presentations.

R. Freedman will attempt to get a copy of the Gould prototype CAIS, and to exercise it on the Polytechnic University computers. He will request support for a graduate student.

D. McCraken will investigate the Testability of CAIS. The investigation will start with the list of CAIS features in the traceability matrix.

G. Taylor and J. Foidl will look at methods of testing for security. They will review the Share test.

B. Abrams will investigate Knowledge Based Systems for generating CAIS tests.

Coordination with E & V

We will try to have more communication with the E & V team. R. Szymansky of E & V has been attending all COMPWG meetings. B. Abrams may attend an E & V meeting.

B. Abrams

KIT MINUTES

MEETING OF 8-10 JULY 1986

SAN FRANCISCO, CALIFORNIA

AGENDA: SEE APPENDIX A

ATTENDEES: SEE APPENDIX B

Tuesday, July 8

1.0 OPENING REMARKS

Tricia Oberndorf, KIT Chairperson, brought the meeting to order. Visitors to the KIT meeting were introduced: Dr. Virgil Cligor, University of Maryland and Les Fraim, Honeywell SCOMP Program Manager, to present perspectives on security implementations; Robbie Hutchinson, MITRE prototyping team, was also present.

2.0 KIT CHAIR REPORT

The CAIS Standardization process is continuing. LOGICON is supporting the comment review process with about 500 comments received. The CAISWG is preparing responses to these comments. The CAIS Editorial Board met in Pittsburgh two weeks ago and will meet again August 4 to prepare the final revision to CAIS Version 1. The Standardization Working Group is scheduled to meet in early October to review the proposed changes for final submission of the Proposed MIL-STD-CAIS.

Jinny Castor, Director, Ada Joint Program Office, will not be able to attend this meeting as originally planned due to the departure of Dr. Ed Leiblein and transition of Sonny Maynard, former VHSIC Program Manager, as the new head of the OSD CSS group. The Ada Validation Policy revision comments are in a review process with a new draft for internal review expected the following week. AJPO is working with NATO to identify joint projects that support the Nunn Amendment to demonstrate interoperability of U.S. and NATO weapon systems. The Amendment has reserved $200 million to support demonstration projects.

The life-cycle support for the CAIS is governed by Department of Defense Instruction 4120.3 defining a tri-service Standardization Board chaired by the AJPO. This group will recommend/not recommend the CAIS as a Military Standard. The comments submitted by Industry during the CAIS Public Review period will be considered by this board as valuable inputs.

Members are reminded that visitors to the KIT meetings must be approved prior to the meeting. U.S. citizens are approved by the KIT Chair and non-U.S. citizens must be approved by the AJPO. Future version of the KIT Public Report may only be available from DTIC.

3.0 WORKING GROUP REPORTS

CAISWG - Clyde Roby reported the CAISWG is continuing responding to the CAIS Comments submitted. The CAISWG expects to complete this process this week. The CAISWG will be continuing progress on the CAIS Rationale and the CAIS Reader's Guide documents but priority will be given to any required support for the CAIS Standardization effort. The goal is to have these documents available in the December-January time frame.

RACWG - Hal Hart reported the RACWG has been responding to comments received on the RAC and generating change proposals to be approved by the KIT during this meeting. Work on the RAC Rationale document will continue after the change proposals are approved.

COMPWG - Bernie Abrams reported the COMPWG is looking at methodologies for security testing and possible applications of Artificial Intelligence in Ada testing. Llyoyd Stiles has generated some Quality Assurance Guidelines for the CAIS.

GACWG - Matt Emerson indicated the GACWG is working on sections of the Interoperability Guide including a new definition of "Interoperability" for review by the KIT. They expect to have additional contractor support during the next quarter for progress on the Transportability Guide.

STONEWG - Ann Reedy reported the STONEWG is focusing more on the CAIS. Their approach is to consider the users perspective and not just the tool writers perspective. STONEWG is attempting to evaluate the appropriateness of the CAIS interfaces through a comparison or analysis of information from the internal interfaces of industry environments. They will try to identify what the distribution and level of these interfaces are as compared to the CAIS interfaces. This may result in future alternatives that may reduce risk in the CAIS evolution.

DEFWG - Judy Kerner reported there have been additions to the Glossary based on new terms defined in the RAC and the Transportability Guide. The term "conforming to the CAIS" will also require explanation.

4.0 EVALUATION AND VALIDATION TEAM REPORT

Ray Szymanski, E&V Chairperson, reviewed current activities of the E&V Team including:

At the AJPO Quarterly Review of the E_&V progress on June 19 he was told "You're dead meat".

The Request for Proposals for the Ada Compiler Evaluation Capability and for the CAIS Validation Capability have been released "TO THE BEST OF MY KNOWLEDGE". Any questions concerning these items are directed to the contracting officer, Lt. Ennulat, at Wright-Patterson AFB.

5.0 GENERAL ANNOUNCEMENTS

The transition to Ada20 still has some problems in the mail services which are being worked. The old "Ada Information" account is still on Ada20. The AJPO is looking at the possibility of having it available on a Personal Computer also for those interested parties that do not have access to the Defense Data Network.

Meetings that may be of interest to KIT personnel include the SIGAda in Pittsburgh 22-25 July, the Future APSE Workshop in Saratoga Springs 10-12 September and the Ada EXPO 86/SIGAda planned for Charleston, West Virginia in November.

Those that are interested in using File Transfer Protocol (FTP) to access data on SIMTEL-20 should be aware that the best time to do this is from 6 P.M. to 8 A.M. when the user load is generally lower. This will help avoid time-outs during the process.

The schedule for future KIT meetings is:

1986	September 22-25	Minneapolis - Honeywell
1987	January 19-22	San Diego - TRW
	April 6-9	Johnson Space Center - NASA
	July (dates TBD)	Los Angeles - TRW
	September (dates TBD)	Washington - ALS/N

The Public Review cycle utilized for CAIS Version 1 is expected to also be used for CAIS Version 2. The exact schedule will be coordinated in the near future. The planned identification to distinguish the CAIS documents is that CAIS Version 1 is designated MIL-STD-CAIS and the subsequent revision in CAIS Version 2 becomes MIL-STD-CAIS-A.

6.0 MORNING BREAK

7.0 TRW REPORT

Frank Belz presented the status of the Security analysis of the CAIS. The CAIS mandatory access control will be based on the Trusted Computing Base as defined in the Orange Book produced by the National Computer Security Center. The question is can the mapping be done from the objects, attributes, processes and relations to the underlying TCB and can it be done appropriately

Regarding CAIS completeness Frank indicated that missing some CAIS interfaces by design will make probably make some tools non-portable but they also make CAIS usable on many additional systems. CAIS Discretionary Security will mandate new tools e.g., program access to data (administrative data) to enact these functions. The process model of the CAIS is more general than the process models of either ASOS or SCOMP and the scope of the changes to ASOS/SCOMP is really not known at this time.

Regarding CAIS Prototyping Frank Tadman reported that node management is nearly complete and attributes are 80% coded. For TEXT_IO they are looking at the MITRE prototype. They are also trying to use multiple interface layers to improve portability.[Regarding the MITRE prototype Rebecca Bowerman said the port from UNIX to VMS was quite uncomplicated since both had validated Ada compilers and the prototype was largely (>90%) in Ada].

Regarding the RAC to CAIS Traceability Study, Judy Kerner reported they have developed a set of key issues mapped to the top ten categories of requirements. The overall results indicate that 63% Substantially or Partially met (48% Substantially; 15% Partially) with 30% in the Deferred status. Only 7% were not in the CAIS specification.

It was noted that after the security mechanisms are implemented a security analysis needs to be performed to insure security requirements are met. Portability is an easier problem to address than performance or security issues.

8.0 BREAK FOR LUNCH

9.0 SOFTECH REPORT

Softech reported they had delivered a Draft Issue Report to NOSC with a final document due in July. While no formal definition of a typing concept is in CAIS 1 it is informally introduced and will be expanded in CAIS 2.

10. ADJOURN FOR WORKING GROUP MEETINGS

11. ADJOURN FOR DAY

Wednesday, July 9

12. ALS/N PRESENTATION

Bill Wilder, ALS/N Program Manager, provided an update on the status of this program. There will be a Public Review of the Ada/M (44) Prototype July 31, from 9 to 12. Those interested in attending should contact SofTech/Washington (703) 931-7372. The ALS/N is primarily constructing a development suite and Run Time Support for the AN/UYK-43/44 series of NAvy computers. The implementation is in Ada and intended to support the development environment & Life Cycle Support activities of the Navy. ALS/N is the start for the standard Navy Software Engineering Environment. In the future they will look if they may be able to implement CAIS as the basis of their environment. They are using the ALS the Army built as the base until then.

13. BREAK FOR NAMED WORKING GROUP

14. SECURITY SESSION

The National Computer Security Center has two organizations specifically concerned with product certification. The C1 group put out the Orange Book and evaluates commercial products for certification. The C2 group is a special project product evaluation team to examine products developed under government contracts.

The certification process follows the following process.

1. Initial product assessment
1.0 The NCSC will assist with development, if necessary
1.1 NCSC advises the vendor on types and format of the evaluation evidence for a particular class of secure system (C1 - A1).
1.2 The NCSC team and vendor produce a schedule for development of evidence.
1.3 Team examines the evidence.
1.4 Team produces Initial Product Assessment Report (IPAR); IPAR justifies the proposed class for formal evaluation.
1.5 Team defends IPAR to the Technical Review Board.
1.6 Management negotiates formal evaluation with vendor.
1.7 Vendor agreement for a specific product in a specific class.

2. Formal Evaluation
2.1 Team/developer produce evaluation schedule
2.2 Team receives training on use of system
2.3 Team evaluates evidence including reading of the product documentation and execution of security test proofs including penetration testing.
2.4 Team provides feedback to developer including any problems/recommendations
2.5 Produces final evaluation report and presents to TRB

B2/B3 Assurance requirements are becoming the standard for quality robust operating systems.

15. BREAK

16. SECURITY SESSION (Cont'd)

It is anticipated that security for development environments will have the following class distribution:

A1	5%
B3	15%
B2	25%
C2	50%

It is expected that eventually there will be a National Directive that all development environments must be at least C2 (with a TCB) required to bid government contracts. MULTICS is at the B2 level. The TCB is 361K lines of code.

The SCOMP is at the A1 level and its TCB is 35K lines of Pascal and C code. In developing a TCB you cannot compromise in either security or compatibility; only on performance.

The anticipated CAIS/TCB issues are expected to occur in the following areas:

access checking/granting
mandatory access must apply to "nodes" "relationships" and attributes" (4.4.3)
user defined access rights
CAIS node growth restrictions
administrator functions
secure attention key

The labelling of nodes (files) in the CAIS model must be handled by the TCB in concert with the Bell-Lapadula axioms.

17. ADJOURN

Thursday, July 10

18. NATO INITIATIVE

LCDR Phil Myers reported a new initiative to develop an APSE is being sponsored by Senator Nunn to encourage collaborative efforts among NATO allies. They have formed a Special Working Group on APSE's as a demonstration project. The objective is to achieve economies in the use of Alliance resources through cooperative efforts to enhance APSE's. The activities include development of a set of software tools on CAIS V1 and to implement CAIS V1 on 2 distinct architectures. There will be an evaluation of individual tools and the APSE's and a continuing analysis of on-going CAIS & PCTE developments. There is an ongoing effort for the definition of requirements for the NATO Interface set (NSIS).

18. BREAK FOR LUNCH

19. WRAPUP

The **CAISWG** reported the CAIS Editorial Board (CEB) is working on CAIS Comments and responses to the received comments.

The **RACWG** is working on unresolved requirements.

The **GACWG** is working on their Transportability Guide but they are also struggling with the Interoperability Guide.

The **COMPWG** is attempting to define what a CAIS conforming implementation means. There is a new Chair of COMPWG, Jack Foidl. They are also obtaining E & V material for review.

STONEWG writing assignments are due 1 August.

Tricia reviewed the Meeting Schedule:

'86	Sep 22 - 25	Minneapolis
'87	Jan 19 - 22	San Diego
	Apr 6 - 9	Houston
	July	TRW/Los Angeles
	Sep	D.C. (ALS/N)

20. BREAK FOR WORKING GROUPS

21. MEETING ADJOURNED

APPENDIX A

Agenda for KIT/KITIA Meeting
7-10 July, 1986
San Francisco, CA

Monday, 7 July:

GACWG 1100-1700
RACWG 1330-1700
STONEWG 1330-1700
No CAISWG meeting
No COMPWG meeting

Tuesday, 8 July:

0800-0830: Arrive and Settle
0830-1030: General Business and Announcements

- Introductions
- Status Reports (contracts, etc.)
- KITIA Chairman Report
- E & V Report
- DIANA Chairman Report
- XWG Chairmen Reports
- Announcements
- Meeting Schedule
- Local Arrangements
- New Business

1030-1045: Break
1045-1145: TRW Report
1145-1315: Lunch
1315-1445: SofTech Report
1445-1500: Break
1500-1700: SofTech Report

Wednesday, 9 July:

0800-0830: ALS ALS/N (Bill Wilder)
0830-1145: Named Working Group Meetings (Break @ chair's discretion)
1145-1315: Lunch
1315-1445: Security Session
1445-1500: Break
1500-1700: Security Session

Thursday, 10 July:

0800-0900: NATO Initiative (Jinny Castor)
0900-1000: RAC Change Proposal Voting
1000-1015: Break
1015-1145: RAC Change Proposal Voting
1145-1315: Lunch
1315-1415: Wrapup
1415-1430: Break
1430-1700: Named Working Groups (Break @ chair's discretion)

APPENDIX B
ATTENDEES

KIT Members

Tony Alden	TRW
Frank Belz	TRW
Matt Emerson	Naval Avionics Center
Jay Ferguson	Department of Defense
Jack Foidl	TRW
Hal Hart	TRW
Robbie Hutchinson	MITRE
Judy Kerner	TRW
Chuck Koch	Naval Air Development Center
Jack Kramer	Institute for Defense Analyses
Sue LeGrand	NASA
Ed McCrohan	U.S. Army Cecom
Gil Myers	Naval Ocean Systems Center
LCDR Philip Myers	Ada Joint Program Office
Tricia Oberndorf	Naval Ocean Systems Center
Shirley Peele	FCDSSA, Dam Neck
Gary Pritchett	SofTech
Dave Pogge	Naval Weapons Center
Clyde Roby	Institute for Defense Analyses
Lloyd Stiles	FCDSSA, San Diego
Ray Szymanski	Wright-Patterson AFB

Frank Tadman	TRW
Jean Tardy	National Defense Headquarters, Canada
Guy Taylor	FCDSSA, Dam Neck
Rich Thall	SofTech
Bill Wilder	NAVSEA/PMS-408

KITIA Members

Bernie Abrams	Grumman Aircraft
Nick Baker	McDonnell Douglas
Dick Drake	IBM
Bob Fainter	Virginia Polytech
Herm Fischer	Litton Data Systems
Ferdinando Gallo	Bull, France
Esa Nurmi	Oy Softplan, Finland
Dianna Peet	Control Data
Erhard Ploedereder	Tartan Labs
Ann Reedy	PRC
Andy Rudmik	GTE Communication Systems
Don Vines	Honeywell

KIT MINUTES

MEETING OF 22-25 SEPTEMBER 1986

MINNEAPOLIS, MINNESOTA

AGENDA: SEE APPENDIX A

ATTENDEES: SEE APPENDIX B

MEETING :IANDOUTS: SEE APPENDIX C

TUESDAY, 23 SEPTEMBER 1986

1.ˀ OPENING REMARKS

1.1 Tricia Oberndorf, KIT Chairperson, brought the meeting to order.

1.2 Tricia introduced new participants present at this meeting. Terry Philips is temporarily replacing Lloyd Stiles for FCDSSA San Diego. Rick McCarthy is the Canadian Ministry of Defense representative replacing Jean Tardy. Additional members of the SofTech design team present are Susan Trager from the Newport office and Shawn Fanning from the San Diego office.

2.0 KIT Chair Report

2.1 The CAIS Standardization process is continuing. Final CAIS Editorial Board meetings are preparing for the presentations to the CAIS Standardization Working Group during the 8-10 October time frame. The final responses to the submitted CAIS comments are about to go on the NET after updating of the Logicon data base.

3.0 Evaluation and Validation Team Report

3.1 Ray Szymanski, E&V Team chairman, reported on the receipt of a baby boy, 9 lbs. 1 oz, 21 1/2 in., at 4:07 A.M. on 16 August named Eric Raymond. Ray indicated this was the first contractual delivery he has received on time. [No mention was made regarding the budget.]

3.2 A formal E&V Team Status Report was not available. The next E&V Team meeting has been scheduled for San Diego to provide a basis for interaction between the E&V Team and KIT contractors.

It is hoped the CAIS Validation Capability contractor is on board and able to support this meeting.

3.3 The CAIS Validation Capability contract should be completed by Wednesday for Ray's review on Friday. The CVC is running about three weeks behind the Ada Compiler Evaluation Capability with an expected award about the third week of November 1986. There were 5 bids received on this procurement.

3.4 Ray expressed his appreciation for the professional approach and consideration shown by the attendees for respecting his non-disclosure status regarding these procurements.

4.0 WORKING GROUP REPORTS

4.1 CAISWG - Clyde Roby reported the CAISWG is continuing responding to the CAIS Comments submitted. All comments but one have been answered. The CAISWG expects to have the CAIS Rationale and the CAIS Reader's Guide documents available for the January meeting for review. Tricia described the naming conventions currently used for the evolving CAIS document. The January 1985 document is referred to as "Proposed MIL-STD-CAIS". The initially standardized document planned for December 1986 standardization will be "DOD-STD-CAIS". It will be a DoD standard since all measurements will also be in metric form (BPCM as well as BPI). The December 1987 version of the CAIS will be "DOD-STD-CAISA". In all versions the "-CAIS" part will be replaced with the official number that is assigned to the document.

4.2 RACWG - Hal Hart reported the RACWG has republished the Requirements and Design Criteria document based on the approved changes from the July meeting. The current version of the document is dated 14 July 1986. Responses to the RAC comments will be uploaded to the NET this month. The Rationale document needs revision to complement the current RAC document.

4.3 GACWG - Matt Emerson indicated the GACWG has an internal draft of the Transportability Guide. Chapters 1, 5 and 6 are to be finalized at this meeting. The Interoperability Guide may become a set of Issue Papers that address specific areas.

4.4 COMPWG - Jack Foidl reported the COMPWG has been asked to draft a definition for "conformance to the CAIS". They hope to accomplish this during this meeting with inputs from Ray Szymanski, the E & V chair, Dr. Lindquist, who is also a member of the E & V Standards Evaluation Working Group, and LCDR Myers of the Ada Joint Program Office.

4.5 STONEWG - Ann Reedy reported the STONEWG is expecting to work on a white paper during this meeting period.

4.6 DEFWG - Judy Kerner reported there have been additions to the Glossary which are available on the NET.

5.0 GENERAL ANNOUNCEMENTS

5.1 Meetings that may be of interest to KIT personnel include the National Conference on Ada Technology, co-sponsored by the Army, scheduled for 16-19 March 1987 in Washington, D.C.. The 9th International Conference on Software Engineering is scheduled for March 1987 in Monterey, California. A NATO Transition to Ada conference is planned for the Hague, Netherlands, on 22-24 October 1986. A Workshop on Commonality in Computing for NASA Flight Systems will occur 28-30 October 1986 at the Johnson Space Center in Houston, Texas. There will also be a Symposium on Practical Software Environments in Palo Alto, California, on December 9-11, 1986.

5.2 The schedule for future KIT meetings is:

1987	January 19-22	San Diego - TRW
	April 6-9	Johnson Space Center - NASA
	July 6-9 or 13-16	San Diego - TRW
	September (dates TBD)	Washington - ALS/N

6 DIANA REPORT

6.1 Rudy Krutar indicated the current support contract for DIANA expires at the end of September. A revised User's Manual is currently under review.

6.2 Rudy expressed his opinion that the AJPO seems to have lost interest in DIANA at this time but he feels it may be relevant to interoperability.

7.0 AJPO STATUS

7.1 LCDR Philip Myers indicated Sebastian Ramos is now on board at the AJPO replacing Paul Cohen as Technical Director.

7.2 Regarding DIANA, the AJPO cannot find serious industry interest for DIANA in the marketplace so AJPO funding is placed on other project areas. If there is meaningful interest in DIANA it is expected the Ada Board should bring this to the AJPO panel.

7.3 The Ada Information Clearinghouse contract is under evaluation at this time.

7.4 Responding to queries regarding Ada Board meetings Philip noted that according to the rules of the Federal Advisory Panel Regulations members of the general public in attendance are observers only and not allowed to actively participate in the meeting.

7.5 Work is progressing on the Ada Compiler Validation Criteria. The AJPO is addressing issues such as the definition of "derived compilers".

7.6 The NATO Initiative should have one more meeting 6 October 1986 before formal signing of the agreement to develop CAIS implementations and compatible tools. The last meeting went very well, even with the lawyers present. It is expected the formal agreement will be signed 22 October 1986. Policy for the export of technology is included in the terms of the agreement.

8.0 TRW REPORT

8.1 Hal Hart presented the status of activities in support of the TRW Ada Software Engineering contract. The CAIS Editorial Board support included completing comment analysis and providing rewrites for the December 86 document (DOD-STD-CAIS). The Transportability Guide activity now has five sections of the document out for review. A final report on the RAC/CAIS Traceability analysis was delivered in July. The RAC has been revised based on the approved change proposals. The CAIS prototyping effort continues at a decreased staffing level and delivery is expected to be provided to NOSC the end of October 1986.

9.0 Morning Break

10.0 SofTech Report

10.1 Due to schedule/flight difficulties experienced by scheduled speakers, the SofTech report was begun early. Rich Thall began the SofTech report by introducing Shawn Fanning and Susan Trager who will be supporting the SofTech design team. At the recent APSE Workshop Rich observed overwhelming agreement with the entity relationship model, with typing. Rich indicated some of the problems associated with distribution is that existing taxonomies deal with wiring and not software.

10.2 General guidelines SofTech applies to distribution were described. The CAIS should be implementable on a heterogeneous distributed system. The database can be spread over the network. The CAIS services must be simple and widely applicable. The main services required are the routing of messages and data to a specific network element, access and storage at a specific element, reception of information from other elements, program execution at a specific element, control of access and concurrent use of elements and examination of the network state.

10.3 Rich also described problems with determination of CAIS boundaries, i.e., is each workstation in a network a single CAIS? SofTech has defined a context for the CAIS operations to address these problems. The model includes a nesting of CAIS contexts whose outer perimeter consists of fixed items which seldom change such as the CAIS Standard or the standard node types. Next come persistent or non volatile items commonly viewed as one CAIS instance such as type definitions. This is followed by the CAIS lifetime consisting of all users of one CAIS instance that are set when the CAIS is initialized. The next context is the session (or job) which is the lifespan of a user authentication such as terminal binding or the process structure. The last context is that of the process which is the lifespan of a tools execution and provides recovery information and I/O bindings.

10.4 From the SofTech perspective, inter-CAIS linkage can be accomplished via explicit import/export operations completed in the lifetime, session or process context. The importation operations provide control over foreign user authorization and authentication, visibility of foreign types, application of concurrency locks, visibility of foreign device drivers, triggers, and serial number generation. Exportation operations establish the portions of the database to be externally visible.

10.5 Inter CAIS relationships are routed through gateway nodes which are visible in the local database. Traversals of relationships "through" the gateway node is mediated by a gatekeeper which is a special sort of device driver which manages communication over the network. Every gateway node to a foreign CAIS has its own gatekeeper. This approach doesn't preclude transparent distribution or telecommunications.

11.0 BREAK FOR LUNCH

12.0 Ontologic Presentation

12.1 Tom Atwood presented the results of the Ontologic object oriented database management systems design for Computer Aided Systems Engineering applications and its compatibility with the Requirements and Criteria (RAC) document requirements for entity management support. Tom provided a comparison of the Ontologic Object Manager currently under development with the specific RAC paragraphs defining "Entity Management Support" defined in RAC Section 4. He also provided copies of his paper on "An Object Oriented DBMS for Design Support Applications".

12.2 An object oriented DBMS provides an excellent framework to model objects, properties and operations. This supports a framework for integrating applications through an interactive interface to link the user into the DBMS.

13.0 SofTech Report (Continues)

13.1 Ed Dunn continued with SofTech's approach to IO and device drivers. The logical device drivers discussed at an earlier meeting has evolved to a common model of IO/Inter Process Communications which involves some parameter changes and an "attribute handle". The IO/IPC services include procedures to OPEN_IO, READ, WRITE, SEND_CONTROL, READ_CONTROL, and CLOSE_IO. The attribute handle is a type to define parameters to support these procedures.

13.2 CAIS_IO services will then support all inter-nodal communications. This allows for definition of standard protocols for certain device classes and encourages driver portability as well as tool portability.

14.0 AFTERNOON BREAK

15.0 SofTech (Cont'd.)

15.1 The discussion on the SofTech typing model includes definition of a Type Definition Language to describe various components of a type including the inheritance, attribute, relationship and constraints associated with a node type. Rules are defined for either generalized or specialized inheritance. A number of issues remain such as how descriptions for attributes (contents) are expressed, what is the impact of importation/exportation, what defines type modification, should types have revisions, should all type names, attribute names and relation names be expressed as pathnames and how are temporal events supported. SofTech appreciates audience inputs on these issues.

15.2 Gary Pritchett presented a brief summary of the CAIS Prototyping. Currently, they have just completed the CAIS 1 prototype detailed design. They are about to implement the Node and Process models and intend to experiment with multiple Ada compilers in the CAIS prototype implementation.

16.0 ADJOURN FOR DAY

WEDNESDAY, 24 SEPTEMBER

17.0 CAIS ISSUE REVIEW

17.1 Tricia Oberndorf described the comment submission and review and response activities for the Proposed Common APSE Interface Set which has been in review for standardization as a Military Standard. A meeting of Department of Defense representatives has been scheduled for the 8-10 October 1986 period to approve/disapprove standardization. All of the 609 received comments on the proposed MIL-STD-CAIS were catalogued and entered into a database maintained by Logicon for distribution, analysis and response by the CAIS Editorial Board (CEB). The CEB responded to all submitted comments, even those received on earlier versions (1.0, 1.1, 2.0 etc.) of the document. As a technical review organization, the CEB did not address those comments that were of a policy nature. Those were forwarded to the Ada Joint Program Office and submitted to the CAIS Standardization Working Group without responses.

17.2 The comments were then organized into logical areas such as Editorial for typo's and grammar improvement, the CAIS physical structure, and the logical CAIS packages. The comments were also evaluated as to their impact, applicability and implementability in the Proposed MIL-STD-CAIS. This allowed the CEB to distinguish difficulties encountered in the proposed document with it's existing scope and those items that were purposely deferred to a later version but against which comments were submitted (i.e., typing, configuration management, distribution, etc.). Those items to be addressed in a subsequent version are so identified in the comment response files and have been provided to SofTech for DOD-STD-CAISA. From this organization an analysis was completed to identify the specific issues raised by the comments.

18.0 BREAK FOR LUNCH

19.0 REORGANIZE INTO WORKING GROUPS

20.0 ADJOURN FOR DAY

THURSDAY, 25 SEPTEMBER

21.0 RAC WORKING GROUP MEETINGS

22.0 MORNING BREAK

23.0 RAC ITEM DISCUSSIONS

23.1 Hal Hart led a lengthy discussion on interoperability, how to define a common external form, and how to transfer and interpret run-time values and floating point data.

24.0 BREAK FOR LUNCH

25.0 KIT WRAP-UP SESSION

25.1 CAISWG does not plan any presentations for the January meeting. The CEB will continue to support the standardization process and hopes to have a published DOD-STD-CAIS by January.

25.2 GACWG will continue work on the Transportability Guide. Jean Tardy has a reorganization they have been working on which should be final by the end of the day.

25.3 COMPWG is looking forward to supporting the E & V Team meeting in San Diego in December. Philip Myers has agreed to provide a copy of the Ada Compiler Validation Criteria policy, when available from AJPO, to assist in formulation of a "CAIS compliance" definition. The COMPWG plans to review the CAIS document from a logical consistency perspective. The intent is to identify that the proposed functionality is logically consistent (each OPEN has a corresponding CLOSE) and the narrative descriptions are unambiguous(particulary in the context of Section 4).

25.4 STONEWG supported the RACWG/CAISWG meetings during this meeting period since they did not have a critical mass of regular supporters.

25.5 DEFWG plans to have a proposed definition of Interoperability on the NET for review and consideration.

25.6 The E & V Team expects to have a revised draft of the CAIS Analysis Document available for the January KIT meeting. This document is the product of their Standards Evaluation Working Group (SEVWG).

26.0 REORGANIZE INTO WORKING GROUPS

27.0 MEETING ADJOURNED

APPENDIX A - Agenda for KIT Meeting
22-25 September, 1986

Monday, 22 September:

Named Working Group meeting as requested by group leaders.

Tuesday, 23 September:

0800-0830: Arrive and Settle
0830-1030: General Business and Announcements

- Introductions
- Status Reports (contracts, etc.)
- E & V Reports
- XWG Chairmen Reports
- Announcements
- Meeting Schedule
- Local Arrangements
- New Business

1030-1045: Break
1045-1130: Object Management System (Tom Atwood, Ontologic)
1130-1145: TRW Report
1145-1315: Lunch
1315-1445: SofTech Report
1445-1500: Break
1500-1700: SofTech Report

Wednesday, 24 September:

0800-0930: CAIS Editorial Board Report - Changes to Proposed MIL-STD CAIS
0930-0945: Breakditorial Board Report - Changes to Proposed MIL-STD CAIS
1145-1315: Lunch
1315-1700: Named Working Group Meetings (Break at chair's discretion)
1930-2200: CAIS Editorial Board - Birds of a Feather

Thursday, 25 September:

0800-0930: RAC Voting and Report from "Tools Integration Strategy" Working Group at Future APSE II Workshop (If time permits)

0930-1145: Numbered Working Group Meetings (Break at Chair's Discretion)

1145-1315: Lunch

1315-1415: AJPO Status Report (Philip Myers) and Wrapup

1415-1430: Break

1430-1700: Named Working Group Meetings

Friday, 26 September:

Environments Panel of the ADA Board

APPENDIX B - MEETING ATTENDEES

KIT MEMBERS

ALDEN, Tony	TRW
DUNN, Ed	SofTech
EMERSON, Matt	NAC
FERGUSON, Jay	DoD
FOIDL, Jack	TRW
HARRISON, Tim	Texas Instruments
HART, Hal	TRW
HOWELL, Chuck	MITRE
HUTCHISON, Robbie	MITRE
KERNER, Judy	TRW
KRUTAR, Rudy	NRL
LEGRANDE, Sue	SofTech
LINDQUIST, Timothy	Arizona State University
MCCARTHY, Rick	National Defense HQ, Canada
MUMM, Hans	NOSC
MUNCK, Bob	MITRE
MURRAY, Margaret	Compusec
MYERS, Philip	AJPO
OBERNDORF, Tricia	NOSC
PHILLIPS, Terry	FCDSSA-SD
POGGE, Dave	NWC
PRITCHETT, Gary	SofTech
ROBY, Clyde	Institute for Defense Analyses
SZYMANSKI, Ray	Wright-Patterson AFB

TAYLOR, Guy	FCDSSA-DN
TEDD, Mike	U.K. MoD
THALL, Rich	SofTech
WOOD, Bill	Software Engineering Institute

Guest Attendees

BAKER, Nick	McDonnell Douglas
DERIUGIN, Tanya	Boeing Aerospace
DRAKE, Dick	IBM
FISCHER, Herm	MARK V
GALLO, Nando	Bull P.C., France
HORTON, Michael	SDC
LYONS, Tim	Software Sciences Ltd., U.K.
MCGONAGLE, Dave	General Electric
REEDY, Ann	PRC
ROUBINE, Olivier	Information Internationale, France
RUDMIK, Andy	GTE Communications
STEVENSON, Bob	Gould
VINES, Don	Honeywell

APPENDIX C - MEETING HANDOUTS

1. KIT Membership List dtd. 09/23/86

2. Guest Directory dtd. 09/23/86

3. KIT/KITIA Minutes, Meeting of 15-17 April 1986, Atlanta, Gecrgia

4. "Ontologic Object Manager and KAPSE dbms requirements", Tom Atwood, Ontologic, 09/18/86

5. "RAC/CAIS Version 1 Compliance Study", TRW for Naval Ocean Systems Center, 30 July 1986

6. "A Distributed CAIS", Sue LeGrand, SofTech Inc., not dated

7. "DoD Requirements and Design Criteria for the Common APSE Interface Set", KIT/KITIA for the Ada Joint Program Office, 14 July 1986

CAIS Implementors Working Group
# of the
Environment Committee

ACM SIGAda Meeting
July 23-25, 1986
Pittsburgh, Pennsylvania

The CAIS Implementors Group met July 23rd, 1986, Wednesday evening, at 7:00 PM. After Clyde Roby, chair, welcomed everyone and took care of some administriva, presentations began.

IBM CAIS IMPLEMENTATION

Jeff Vermette of IBM delivered a presentation of their implementation of the CAIS (a partial implementation). A copy of the slides that were handed out at the meeting will be mailed out to everyone on the mailing list. Some of the more interesting points are discussed below.

Performance of the CAIS implementation was about the same as IBM's CMS. Not all of the CAIS was implemented, just a "reasonable" subset. The development effort was from June 1985 to early 1986 and was done by Jeff with occasional help by one or two others. Not all of the interfaces were implemented; those that were are the ones that have a parameter of NODE_TYPE rather than a parameter of NAME_STRING.

IBM's implementation did not include: mandatory access control (discretionary access control was implemented via attributes), process spawning (can't do under VM -- but was simulated via attributes), input/output packages (but a package was written that translated CAIS I/O to Ada I/O), PAGE_TERMINAL and SCROLL_TERMINAL packages.

Added interfaces included the aforementioned set of packages between CAIS I/O and Ada I/O and a package CAIS_ADDITIONAL_INTERFACES for backup and restore. Backup was done either: explicity, every n increments of time, or every n transactions to the database. This backup would take 30 seconds to a minute depending upon the size of the CAIS database at the time.

The CLI was developed to exercise all of the interfaces as well as initiate some tools. All of the tools were rather simple (caring about an input file and an output file); no tools took real advantage of the CAIS node model. The file structure was hierarchical under the user node. In a multi-user environment, each user had his own virtual machine. On some tools, the CLI exported the file, ran the tool, and then imported the resulting file back to the CAIS. The host tool, in these cases, did not know about the CAIS node structure.

The final subset version of the CAIS was entirely memory resident.

Discretionary access control on the multi-user version was (NONE, LOW, MEDIUM, HIGH, SYSTEM) which corresonded roughly to (<empty>, EXISTENCE, READ, READ and WRITE, all) in the CAIS; this was implemented via attributes.

For the last three months of the project, the CAIS was used as the development environment. The difference in execution speed between a tool running on the CAIS compared with it running on CMS was in the millisecond level (negligible).

Dick Drake of IBM noted that, in general, in early implementations of the CAIS, many tools will be imported from existing vendor environments and will be done as described above.

Normally, there were about 300-400 nodes in the CAIS database; it went as high as about 1200 (which is greater than can be in memory at one time). The parameters chosen in this particular implementation for automatic backup was every 25 write operations or every 15 minutes. 1000 nodes were cached in memory; each node record was about 70-80 bytes (for one to five attributes); type NODE_TYPE took about 50-55 bytes.

GOULD PRESENTATION

Jose Alea of Gould gave a short presentation on what they have been doing. Gould held a CAIS Symposium in mid-April where they introduced their CAIS implementation and gave a demonstration. Everything is implemented but FORM_TERMINAL and MAGNETIC_TAPE. At that time, their distribution policy was announced; this is basically $500 for a single-site license.

Gould is preparing a GOULD CAIS PROTOTYPE USERS GUIDE. It explains the features of their prototype and how to use it. It is not a guide for use of the CAIS in general, though. A copy of their USERS GUIDE will be sent to anyone who purchases a license for use of their prototype.

Gould has also prepared position papers on Exceptions and Iterators for the KIT's CAIS Working Group (CAISWG). They will publish their Technical Paper in a forthcoming AdaLetters.

In the future, Gould is planning to make their prototype functionally complete (they will complete the FORM_TERMINAL and MAGNETIC_TAPE packages), to make it conform with the DOD-STD-CAIS, and to enhance the performance of their implementation as well as improve the multi-user environment. Internal priorities at Gould will determine what actually gets done.

TRW and SOFTECH WORK

Hal Hart of TRW described the current status of TRW's CAIS prototyping & other work as support contractor to NOSC/AJPO for the KIT/KITIA, supporting CAIS specification & standardization. He described how TRW's work complements SofTech's, who is contracted to specify DOD-STD-CAIS-A, for which an initial specification is due in January 1987 with a final ready for submission into the Military Standardization process one year later; SofTech's tasking also includes development of CAIS prototypes.

TRW's CAIS prototype effort will, in the near-term, emphasize "appropriateness" investigations of the current CAIS. Their results will be inputs to possibly influence specifications for DOD-STD-CAIS-A. Appropriate investigations will include tool-building experiments and instrumentation (and possibly simulation) atop a CAIS prototype to:

1) ascertain areas of inconvenience to tool builders, with recommendations for alleviation, e.g., development of standard tool-builder &/or administration tools (with CAIS access control mechanisms being a likely area of such concerns) or changing "levels" of certain interface specifications [see next bullet], and

2) identify likely efficiency impediments in the CAIS specification, e.g., wrong "levels" of interfaces:

   * same sequences of several interfaces being called frequently when their replacement with a higher-level combined-functionality interface could be more efficiently implemented, or

   * too-high-level interfaces where too much functionality is included in one interface, with all the functionality rarely needed at once from typical tools -- meaning several interfaces ought to replace the one.

In addition, Hal will be provided the address list of the CAIS Implementors Working Group so that he can send them the new Bastille Day version of the RAC document.

INTERMETRICS WORK

Mike Ryer of Intermetrics told the group that they have developed a CAIS subset to use in their Ada Program Library (the node model). They hava used the file and structure nodes, attributes, relations and a few other things. Implementations of this have been ported to DEC VMS, Sperry 1100 OS, IBM CMS and IBM MVS. They have achieved portability across these machines/OS's. They either use the host's ISAM or write the equivalent in Ada for the node database.

CHANGES TO PROPOSED DOD-STD-CAIS

Clyde Roby then mentioned some of the changes that are currently being discussed as a result of formal and informal comments against the proposed DOD-STD-CAIS. These include: "flattening out" the CAIS package structure, giving appropriate names to the interfaces (a naming convention has been defined and used), expanding USE_ERROR and NAME_ERROR into more exceptions, changes to LIST_UTILITIES, and clarifying semantics in several areas of the document, especially the section on security.

During this meeting, Jeff Vermette of IBM raised a question concerning the order of checking for and raising exceptions. This generated some discussion, mainly with Erhard Ploedereder of the CAISWG. Jeff will generate a short paper to be submitted to CAIS-COMMENT and CAISWG for further discussion.

CAIS STANDARDIZATION PROCESS

The CAIS Standardization process schedule was presented. The Stanardization Working Group, the body that will decide if the CAIS should be a military standard, will meet in October to vote. It was brought out that there is a possibility that there may be another review cycle before the CAIS is finally standardized.

CAIS RELATED DOCUMENTS

Additional CAIS support documents that are in the process of being prepared include a Requirements and Criteria document (RAC) for the DOD-STD-CAIS-A and a Guidelines and Conventions document. The latter document is a combination of an Interoperability Guide and a Transportability Guide. In addition, two documents for the proposed DOD-STD-CAIS, a CAIS Readers' Guide and the CAIS Rationale document, are nearing completion. It is expected that the CAIS Rationale document for the proposed DOD-STD-CAIS will be finalized within three months of the adoption of the CAIS as a military standard. As was mentioned earlier, Gould is producing their own CAIS Users' Guide (specific to their implementation).

CAIS TOOL REPOSITORY

It was also mentioned that a CAIS tool repository would be started on Ada20 and that it would be handled by the KIT Support contractor. No further details were given.

ADA AND SPACE STATION CONFERENCE PROCEEDINGS

Since the last CAIS Implementors WG meeting, NASA in conjunction with the University of Houston at Clear Lake held a conference about "Ada and the Space Station". Dick Kessinger was the Technical Chairman and has offered to process any requests for proceedings. Anyone who would like to order these proceedings should request them from Dick Kessinger of SofTech at the following address. The cost is $25 per set of proceedings.

Dick Kessinger
SofTech
Suite 105
1300 Hercules Drive
Houston, TX 77058
(713) 480-1994

NOVEMBER MEETING IN CHARLESTON

We are looking forward to the presentations by TRW and Intermetrics at the November meeting of the CAIS Implementors Group. In addition, there will probably be a CAIS Panel during one of the regular day sessions of the Environment Committee. More on this will be distributed as details are worked out.

Section 3

KIT/KITIA DOCUMENTATION

Ada Programming Support Environment

(APSE)

Interoperability and Transportability (I&T)

Management Plan

January 1986

Contract # N66001-86-D-0156

Delivery Order 0001
CDRL Item A019AA

for
NAVAL OCEAN SYSTEMS CENTER
271 Catalina Boulevard
San Diego, California 92152

prepared by

TRW Defense Systems Group
3420 Kenyon Street
San Diego, California 92110

Ada Programming Support Environment

(APSE)

Interoperability and Transportability (I&T)

Management Plan

January 1986

for

Ada JOINT PROGRAM OFFICE
The Pentagon
Washington, D.C. 20301

prepared by

NAVAL OCEAN SYSTEMS CENTER
271 Catalina Boulevard
San Diego, California 92152

TABLE OF CONTENTS

1.0 INTRODUCTION

The Ada Programming Support Environment (APSE) Interoperability and Transportability (I&T) Plan is presented in this document. The I&T activities necessary to achieve sharing of tools and data bases between APSEs are described. Schedules and milestones for these activities are presented as well as a Work Breakdown Structure (WBS) for accomplishing them.

These I&T activities are conducted by the Kernel APSE Interface Team (KIT).

The major responsibilities are:

a. APSE I&T Management
b. APSE I&T Analysis
c. APSE I&T Standards Development
d. APSE I&T Tools Development
e. APSE I&T Coordination with Implementation Efforts

1.1 BACKGROUND

In 1975 the High Order Language Working Group (HOLWG) was formed under the auspices of the U.S. Department of Defense (DoD) with the goal of establishing a single high order language for new DoD Embedded Computer Systems (ECS). The technical requirements for the common language were finalized in the Steelman report [1] of June 1978. International competition was used to select the new common language design. In 1979 the DoD selected the design developed by Jean Ichbiah and his colleagues at CII-Honeywell Bull. The language was named Ada in honor of Augusta Ada Byron (1816-1851), the daughter of Lord Byron and the first computer programmer.

It was realized early in the development process that acceptance of a common language and the benefits derived from a common language could be increased substantially by the development of an integrated system of software development and maintenance tools. The requirements for such an Ada programming environment were stated in the STONEMAN document [2]. The STONEMAN paints a broad picture of the needs and identifies the relationships of the parts of an integrated APSE. STONEMAN identifies the APSE as support for "the development and maintenance of Ada application software throughout its life cycle". The APSE is to provide a well-coordinated set of tools with uniform interfaces to support a programming project throughout its life cycle. The Initial Operational Capabilities (IOCs) are called Minimal Ada Programming Support Environments (MAPSEs).

[1] Requirements For High Order Computer Programming Languages: STEELMAN, DoD, June 1978

[2] Requirements for Ada Programming Support Environments, STONEMAN, DoD, February 1980

The Army and Air Force began separate developments of APSEs. The Army APSE has been designated the ALS (Ada Language System) and that of the Air Force, the AIE (Ada Integrated Environment). The Navy APSE will make maximum use of those Army and Air Force products that meet Navy requirements and will require the development of only those additional components required for Navy applications.

The Ada Joint Program Office (AJPO) was formed in December 1980. The AJPO coordinates all Ada efforts within DoD to ensure their compatibility with the requirements of other Services and DoD agencies, to avoid duplicative efforts, and to maximize sharing of resources. The AJPO is the principal DoD agent for development, support and distribution of Ada tools and Ada common libraries.

1.2 DEFINITIONS

INTEROPERABILITY: Interoperability is the ability of APSEs to exchange data base objects and their relationships in forms usable by tools and user programs without conversion. Interoperability is measured in the degree to which this exchange can be accomplished without conversion.

TRANSPORTABILITY: Transportability of an APSE tool is the ability of the tool to be installed on a different KAPSE; the tool must perform with the same functionality in both APSEs. Transportability is measured in the degree to which this installation can be accomplished without reprogramming. Portability and transferability are commonly used synonyms.

1.3 OBJECTIVES

The objectives of the APSE I&T effort are:

1. To develop requirements for APSE I&T.

   STONEMAN paints a broad picture of the needs and relationships of the parts of an integrated APSE. Although STONEMAN is being used as the primary requirements document for APSE development efforts, it does not provide sufficient detail to assure I&T between APSEs. APSEs built to accomodate I&T requirements will insure cost savings in the development of tools. The cost of reprogramming tools for different APSEs will be significantly reduced.

2. To develop guidelines, conventions and standards to be used to achieve I&T of APSEs.

   Guidelines, conventions, and standards describe the means by which the requirements can be satisfied. There is little precedent for I&T between programming support environments of this anticipated magnitude and thus little guidance for the development of these guidelines, conventions, and standards. The guidelines, conventions and standards that are developed during this APSE I&T effort are evolving over a five year period from 1982 through 1987. These guidelines, conventions, and standards are presented in public forums to insure that they are sound and realistic.

3. To develop APSE I&T tools to be integrated into APSEs.

This APSE I&T effort provides for the development of tools to be integrated into various APSEs. These tool development efforts will help identify interfaces and surface interface problems associated with I&T between different APSEs. They should also show how closely the guidelines, conventions and standards developed by this APSE I&T effort reflect the reality of the various APSE efforts. But the tools developed by this APSE I&T effort will not be limited to this test function. They will also be well-documented tools which will become useful additions to any APSE.

4. To monitor the AIE, ALS, and ALS/N development efforts with respect to APSE I&T.

This APSE I&T effort provides for the monitoring of the AIE, ALS and ALS/N development efforts. The monitoring will result in recommendations for resolution of differences between the AIE, ALS or the ALS/N and the evolving APSE I&T conventions and standards. Interface areas which would inhibit I&T between the APSEs will also be identified.

5. To provide initiative and give a focal point with respect to APSE I&T.

A focal point is needed for APSE developers and users with regard to information about I&T. APSE I&T questions arise frequently within professional societies and user groups. A forum is needed in which APSE I&T questions can be addressed and discussed and in which APSE I&T information can be disseminated throughout the Ada community.

The KIT and KITIA (see Sections 2.3 and 2.4) will provide focal points for the Ada community. Public reports on the results of this APSE I&T effort will be published every six months. This is in keeping with the AJPO philosophy of public exposure of all aspects of the Ada program. The KIT and KITIA will also participate in other programs connected with APSE I&T, including international development efforts, whenever possible.

6. To develop and implement procedures to determine compliance of APSE developments with APSE I&T requirements, guidelines, conventions and standards.

Procedures must be established by which the recommendations that are developed by this APSE I&T effort will be reviewed and implemented by the AJPO. The procedures that are to be followed should apply not only to the AIE and ALS development efforts, but also to other APSE development efforts. Work on the determination of compliance procedures will be pursued in cooperation with the AJPO's

Evaluation and Validation program.

1.4 DOCUMENT ORGANIZATION

Section 1 of this document discusses the purpose and scope of the I&T Plan, the objectives of the I&T effort, and the basic concepts, definitions, and objectives.

Section 2 discusses the sponsorship, the participating organizations, the organizational inter-relationships and responsibilities, and the potential forums for public involvement.

The specific tasks to be accomplished in pursuit of I&T are covered in Section 3. These functions are presented in a work breakdown structure for the project and a schedule of milestones and deliverables.

Special needs in achieving I&T are discussed in Section 4, and a list of references is given in Section 5.

Appendix A contains a glossary of terms and acronyms applicable to the I&T effort. Appendix B describes the elements of the I&T Work Breakdown Structure.

2.0 ORGANIZATION

Figure 1 shows the participants in the APSE I&T effort. The following sections provide a brief description of these organizations and their relationships.

2.1 Ada JOINT PROGRAM OFFICE

The KIT is an agent of the Ada Joint Program Office (AJPO). The KIT supports the AJPO by performing the activities outlined in this plan and by providing recommendations and information to the AJPO. The AJPO makes final decisions in the areas of requirements, policy, procedures and funding.

The Software Technology for Adaptable Reliable Systems (STARS) Joint Program Office (SJPO) was formed after the initiation of the KIT effort. It has come to take an increasing responsibility for efforts involving software engineering environments (SEEs). The STARS SEE projects and the KIT/KITIA have some co-members and share and review one another's documents. Official liaison between the two groups is through the joint program offices.

2.2 ARMY, AIR FORCE AND NAVY

Currently the Army and Air Force have begun separate developments of APSEs. In the development of its APSE, the Navy plans to make maximum use of Army/Air Force products that meet Navy requirements. The KIT will review of all these APSE developments and identify critical aspects of the designs where conventions or standard interfaces and specifications are needed to insure compatibility. It will be the role of the KIT to interact with these services and their respective APSE contractors for information-exchange and consultation. The contractor for the Army's ALS is SofTech Inc.; the Air Force contractor for the AIE is Intermetrics Inc.. The Navy contractor is Control Data Corporation. Representatives of both the Air Force and Army APSE development efforts are members of the KIT, and many members of the Navy's Design Review Group (DRG) serve on the KIT as well.

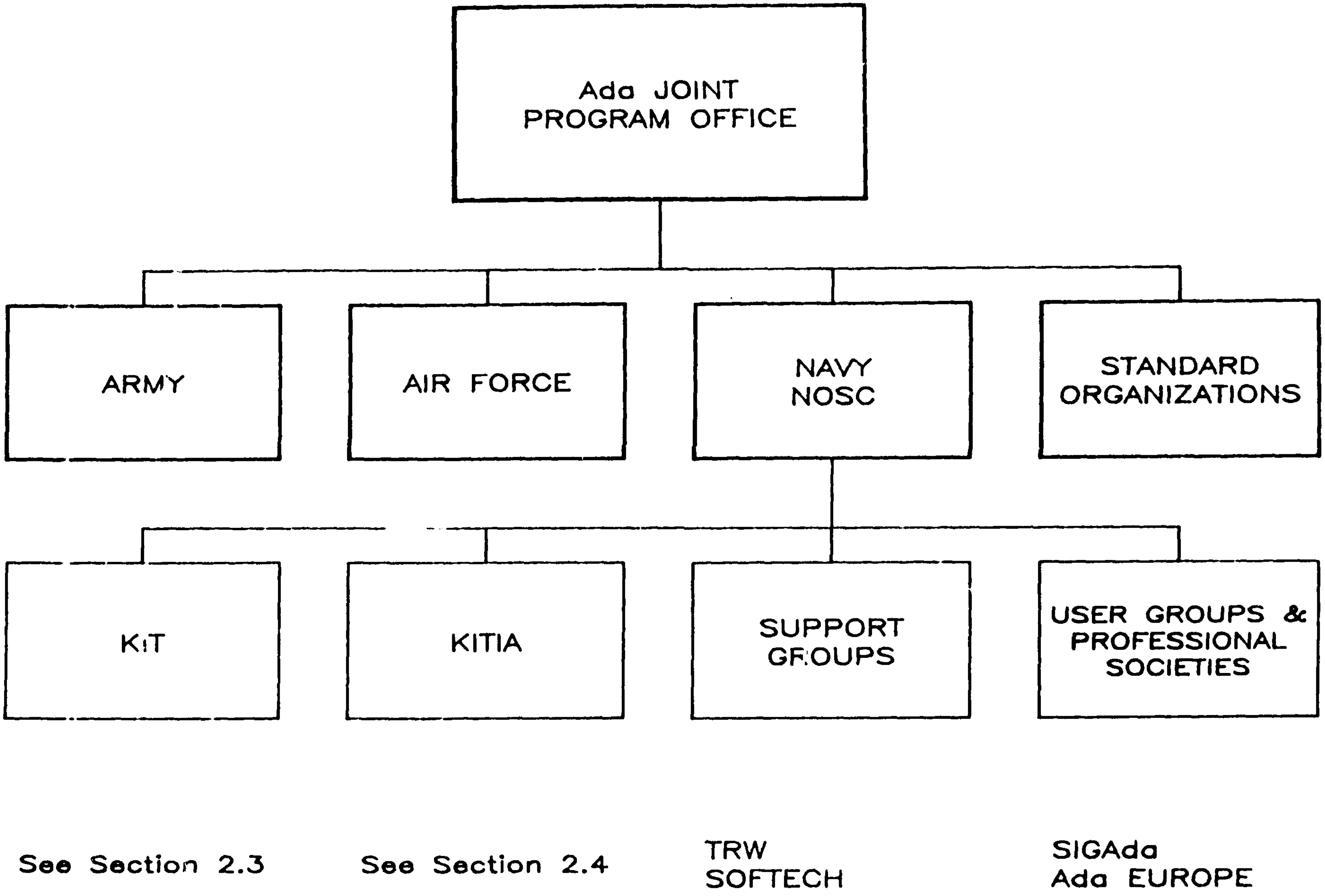


Figure 1. APSE I&T Participants

2.3 KAPSE INTERFACE TEAM (KIT)

The objectives of the KIT are the objectives of the APSE I&T effort (see Section 1.3). The Navy is responsible for chairing the KIT. The membership is composed of the following DoD representatives:

- Navy Deputy to the Ada Joint Program Office
- Naval Ocean Systems Center (NOSC)
- Naval Sea Systems Command (NAVSEA/PMS-408)
- Naval Space and Warfare Systems Command (SPAWAR)
- Naval Underwater Systems Center (NUSC)
- Naval Avionics Center (NAC)
- Naval Air Development Center (NADC)
- Naval Research Laboratory (NRL)
- Naval Weapons Center (NWC)
- Fleet Combat Direction System Support Activity (FCDSSA) - Dam Neck
- Fleet Combat Direction System Support Activity (FCDSSA) - San Diego
- U.S. Air Force - Rome Air Development Center (RADC)
- U.S. Air Force - HQ ISSC/SIC
- U.S. Army - Communications and Electronics Command
- National Security Agency

In addition to the Department of Defense representatives, a number of interested organizations and companies providing contract support are also members of the KIT including:

- Canadian National Defense Headquarters
- Institute for Defense Analyses (IDA)
- National Aeronautics and Space Administration (NASA)
- Software Engineering Institute (SEI)
- United Kingdom Ministry of Defence (MOD)
- MITRE Corporation
- ORACLE Corporation
- SofTech, Incorporated (CAIS Version 2 Design Contract)
- TRW Incorporated (KIT Support Contract)

NOSC is the Navy laboratory which has assumed the responsibility of the KIT chairman. All other members participate on a volunteer basis, aided as necessary by the AJPO with funding for such things as travel expenses. New members will be added to the KIT at the discretion of the AJPO.

In addition, the KIT is divided into various working groups for the purpose of small group concentration on specific technical areas affecting I&T. The number, activities, and membership of such working groups may change as KIT needs change. Currently, the following working groups are active:

- Common APSE Interface Set Working Group (CAISWG) - The objective of this working group has been to design the initial set of KAPSE-level interfaces. They are also responsible for fielding and answering the comments submitted by reviewers of the designed set. The CAISWG will have primary responsibility for technical evaluation of the CAIS Version 2 contractor's products. The CAISWG will also provide recommendations to the AJPO CAIS Control Board for future CAIS activities.

- Requirements and Design Critieria Working Group (RACWG) - The objective of this working group is to define and refine the requirements to which CAIS Version 2 is to be designed.

- Guidelines and Conventions Working Group (GACWG) - The objective of this working group is to define the various guidelines for writing transportable software, in addition to the use of the CAIS.

- Compliance Working Group (COMPWG) - The objective of this working group is to consider the CAIS and other KIT products from the standpoint of determination of implementation conformance.

- STONEMAN Working Group (STONEWG) - The objective of this working group is to refine the current STONEMAN document to make it more useful and usable as a guiding document for all KIT activities.

- Definitions Working Group (DEFWG) - The objective of this working group is to maintain consistency in the use of terms in all KIT documents and related activities.

2.4 KAPSE INTERFACE TEAM FROM INDUSTRY AND ACADEMIA

The KITIA was formed to complement the KIT and to generally contribute a non-DoD perspective to the I&T effort. The KITIA supplements the activities of the KIT. It assures broad inputs from software experts and eventual users of APSE's. The KITIA interacts with the KIT as reviewers, as proposers of APSE I&T requirements, guidelines, conventions and standards, and as consultants concerning implementation implications. The team was selected from applicants representing industry and academia. The following are the members of the KITIA:

Alpha-Omega Group
Aerospace Corporation
Arizona State University
Bell Laboratories
Boeing Aerospace

Bull
Computer Sciences Corporation
Control Data Corporation
Commission of the Europen Communities
Ford Aerospace
Frey Federal Systems
General Electric
General Telephone & Electronics Laboratories
Georgia Institute of Technology
Grumman Aerospace
Hazeltine
Honeywell
Hughes Aircraft
IABG (W. Germany)
IBM
Information Internationale
Litton
Lockheed
McDonnell Douglas
Norden
PRC
Raytheon
SDC
Texas Instruments
UK Ada Consortium
Virginia Polytechnic Institute

In addition, the following is a special associate member of the team:

Oy Softplan Ab (Finland)

Membership on the team belongs to a company or university and not to an individual representing his/her organization. All participation is voluntary, and the members selected have agreed to provide 1/3 of a man-year plus other support such as travel expenses. The membership of the KITIA will not be expanded unless an organization withdraws or very special circumstances apply. The AJPO sponsor and KIT chairman are ex-officio members of the KITIA.

The KITIA elects a chairman from amongst its participants every year. It is organized into groups who in turn select their own chairmen. The KITIA chairman together with the group chairmen form the KITIA management committee. In addition, the KITIA is divided into the same set of working groups as the KIT (see Section 2.3).

The KITIA is responsible to the AJPO through the KIT chairman. Although the KIT has ultimate responsibility for the development of all products required to meet the I&T objectives, the KITIA participates directly in the generation and review of such products. In addition, the KITIA generates its own contributing papers, products, initiatives, and recommendations to supplement and guide the basic KIT efforts. This requires close coordination, which is facilitated by ARPANET/MILNET communication mechanisms, parallel working group structures, and joint team meetings.

2.5 SUPPORT CONTRACTORS

Currently there are two NOSC contractors that participate on the KIT. TRW is the primary support contractor, providing general support and technical initiatives. The SofTech/Compusec team is under contract to evolve the initial work of the KIT/KITIA CAIS Working Group for development of a Common APSE Interface Set Version 2 which is to be submitted for approval as a military standard.

Through their technical support contract to the Ada Joint Program Office and their continuing support to the CAIS development effort, the Institute for Defense Analyses also participates as a member of the KIT.

2.6 EVALUATION AND VALIDATION (E&V) TEAM

In keeping with the approach to Ada itself, the AJPO intends to be able to validate conformance of implementations to the CAIS. Criteria for such validation testing of the CAIS is the concern of the COMPWG (see Section 2.3), working closely with the Air Force-lead Evaluation and Validation Team. The Air Force Wright Aeronautical Laboratories is the lead organization of the Evaluation and Validation Team which is developing the technology required for a central agent to perform I&T validation testing on each APSE. The models for a strong central validation capability are the Ada Compiler Validation Capability and the Ada Validation Organization (AVO).

2.7 USER GROUPS AND PROFESSIONAL SOCIETIES

SIGAda, the Ada-JOVIAL Users Group (Ada-JUG), and Ada Europe provide valuable contributions to the APSE I&T effort. The KIT and KITIA have no formal relationship with these groups; however, the AJPO will use some or all of these groups as regular forums for the presentation of reports and technical results and will solicit feedback from their members. In addition, the CAIS Implementors Group (CIG) (see Section 2.9) is a sub-group of SIGAda. Another SIGAda sponsored sub-group that is expected to provide additional technical inputs is the Ada Run-Time Environments Working Group (ARTEWG).

2.8 STANDARDS ORGANIZATIONS

The American National Standards Institute (ANSI) and the International Standards Organization (ISO) are standards organizations which are already involved in establishing the Ada programming language as a broadly recognized, enforceable standard. It is possible that the results of this I&T effort will be submitted for such approval by these organizations as well, to effect the commonality of APSE's deemed necessary to achieve DoD's life-cycle objectives. The KIT initially will become familiar with the organizations' standardization procedures so that future standardization actions can be planned and accomplished with minimum difficulty. This will include the study of existing standards which may interact with or guide the development of APSE I&T standards.

2.9 LIAISON WITH IMPLEMENTATION EFFORTS

Through their participation as members of the KIT, the DoD APSE implementations of the AIE, ALS, and ALS/N environments provide status and technical interchange for continuing insight into I&T related issues.

A number of APSE implementation efforts have been undertaken by organizations outside of the DoD. Three of these (the U.K. Ada Consortium, the West German IABG and U.C. Irvine) have participated on the KITIA. Others include the European Economic Community, ROLM Corporation, Western Digital, and Telesoft, just to name a few. The KIT will keep such organizations informed of its activities and will consider all feedback received from them.

In addition, a number of organizations are currently involved, to varying degrees, in implementations of the first version of the CAIS. These organizations have formed a CAIS Implementor's Group (CIG) which meets during SIGAda meetings. Members of the KIT/KITIA CAISWG participate in the CIG and provide liaison between the groups.

3.0 APSE I&T PLAN

This section shows the Work Breakdown Structure (WBS) for the I&T effort as well as the schedules and milestones for the WBS elements. Figures 2 thru 7 provide an overview for the WBS elements.

3.1 WORK BREAKDOWN STRUCTURE

A discussion of the major elements in the WBS is presented below. Detailed task descriptions are contained in Appendix B.

1000 APSE Interoperability & Transportability Management

This WBS element covers the general management tasks required to accomplish the APSE I&T objectives. It includes general project and team management, project planning, general meeting and team support and configuration management.

2000 APSE Interoperability & Transportability Analysis

This WBS element covers the technical analysis tasks required to accomplish the APSE I&T objectives. It includes resource reviews, requirements development, and performance of special studies.

3000 APSE Interoperability & Transportability Standards

This WBS element describes the standardization tasks required to accomplish the APSE I&T objectives. It includes guidelines and conventions development, specification development, compliance and validation formulation, common APSE interface set analysis, and definition and support of the standardization process.

4000 APSE Interoperability & Transportability Tools

This WBS element describes the development of APSE tools that support the APSE I&T objectives. It includes planning and acquisition of tools, tool development, test and analysis, and maintenance and modification of developed tools.

5000 APSE Coordination with Implementation Efforts

This WBS element describes the tasks affecting various APSE development efforts required to support the APSE I&T objectives. It includes public reviews of the AIE and ALS, development of prototypes of the Common APSE Interface Set, I&T analysis of AIE and ALS, and liaison with other implementations groups.

WBS Overview

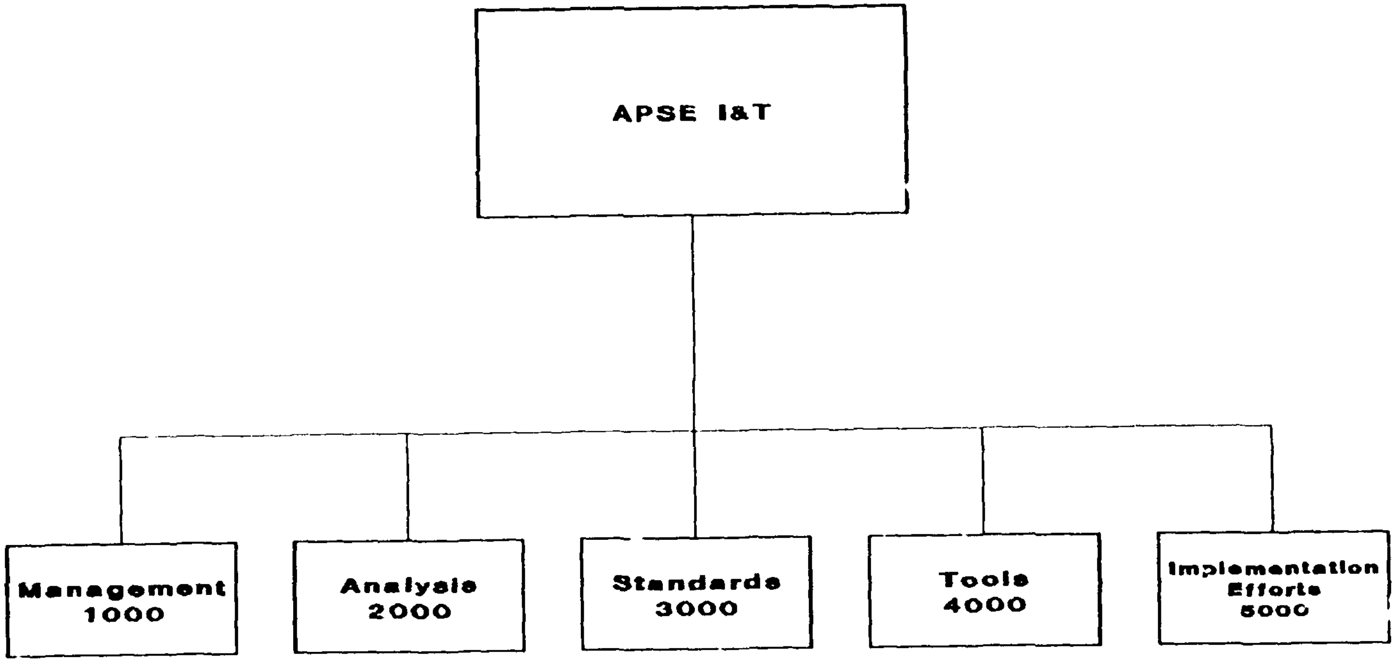


Figure 2. WBS Overview

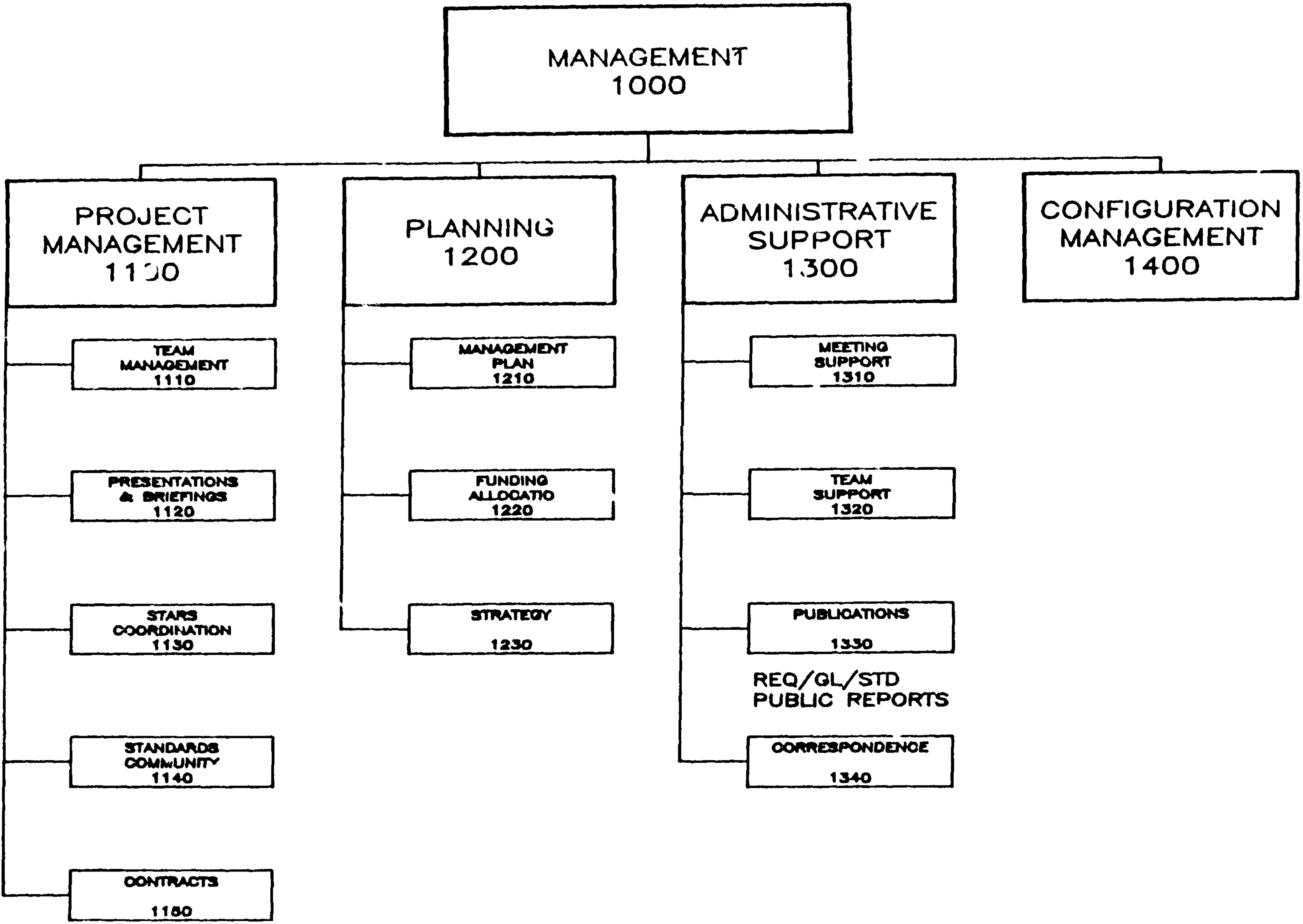


Figure 3 WBS Management Overview

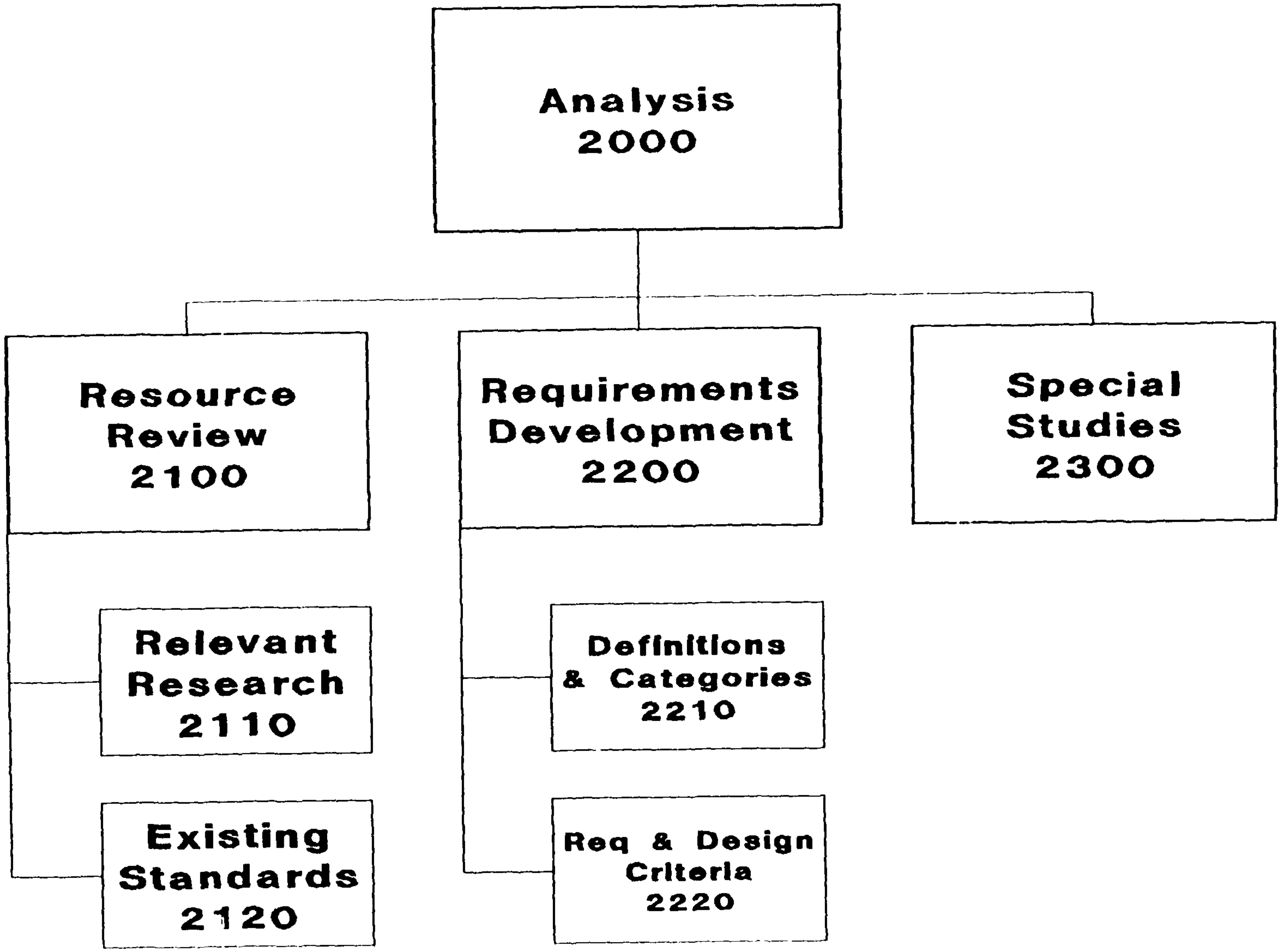


Figure 4. WBS Analysis Overview

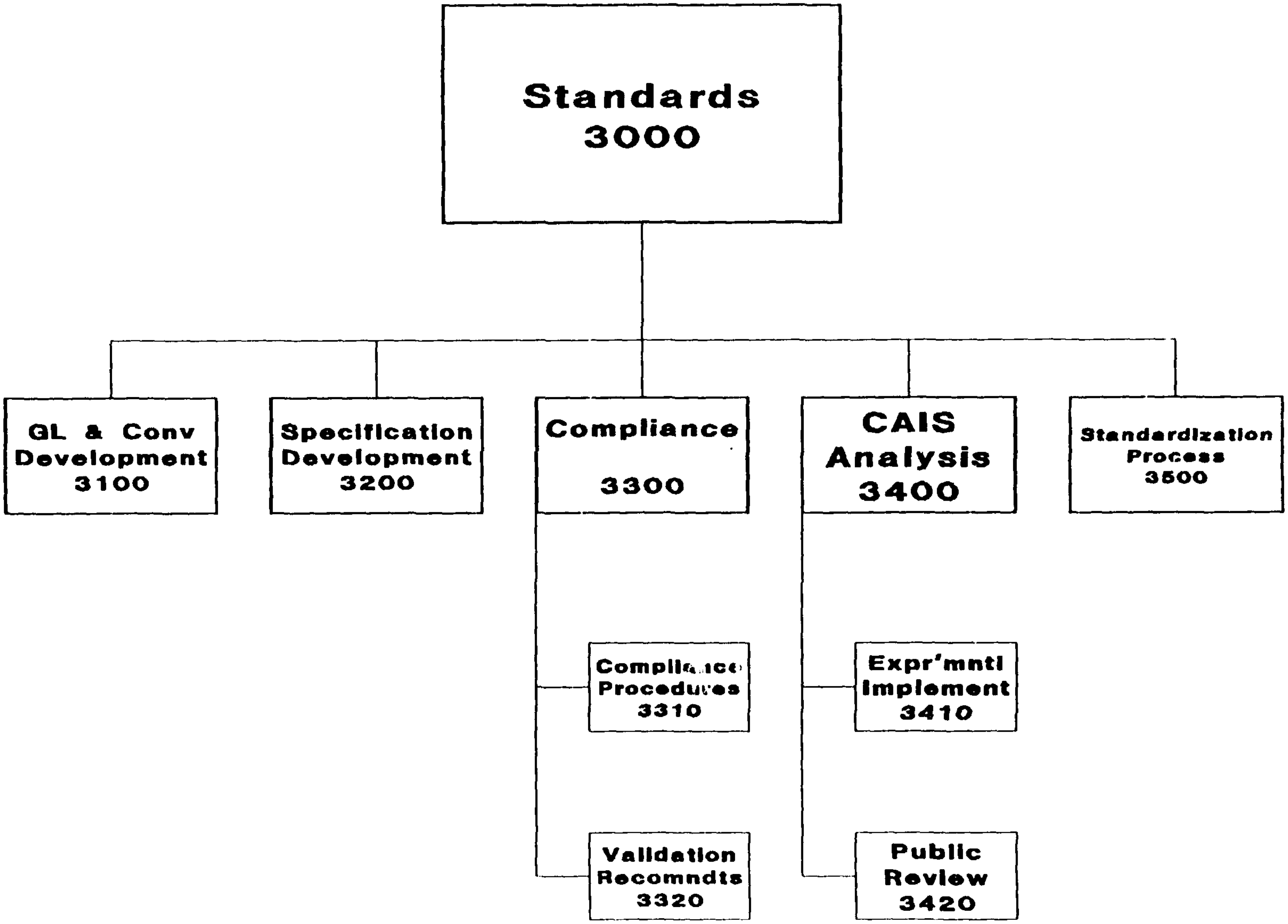


Figure 5. WBS Standards Overview

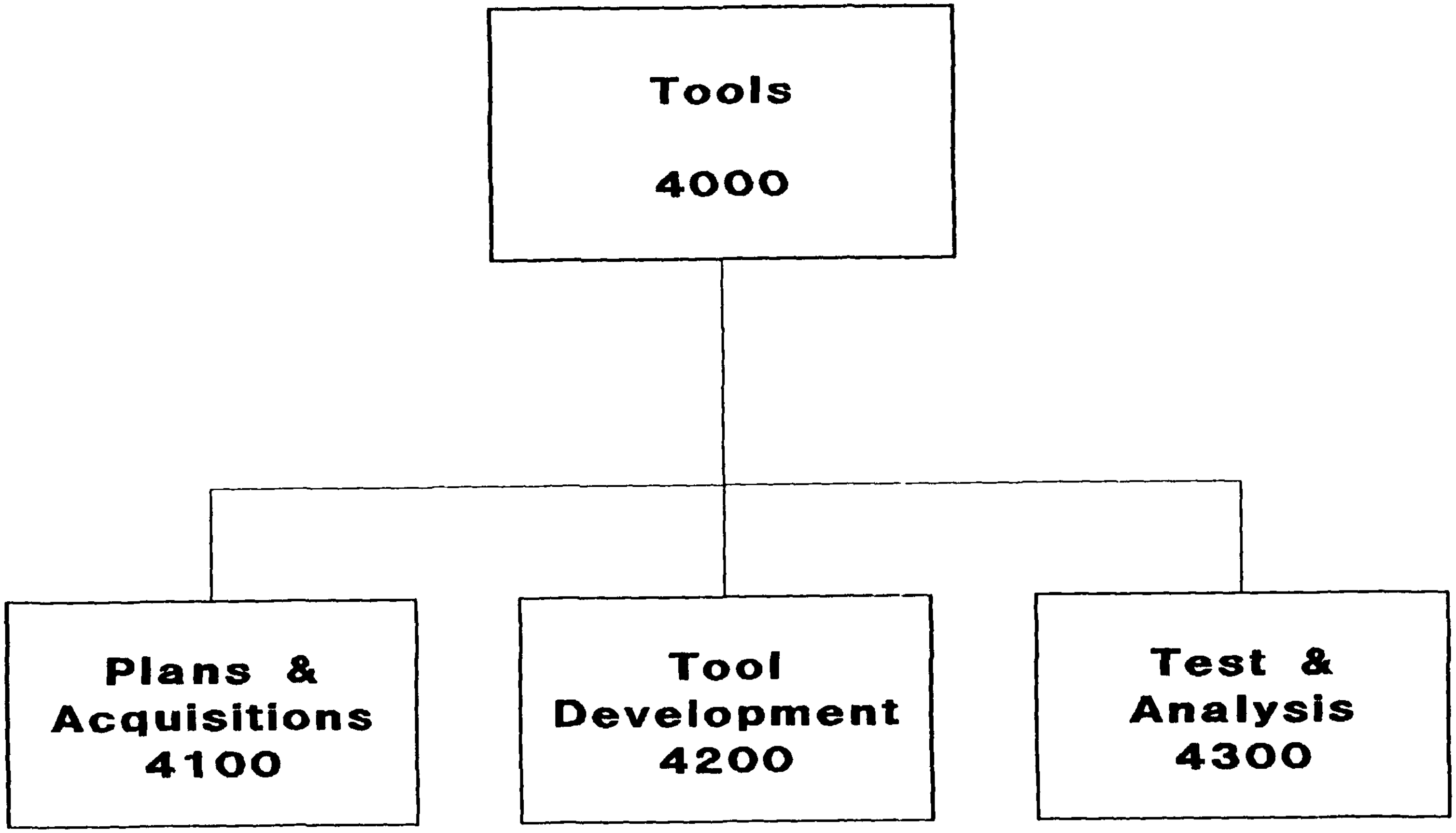


Figure 6. WBS Tools Overview

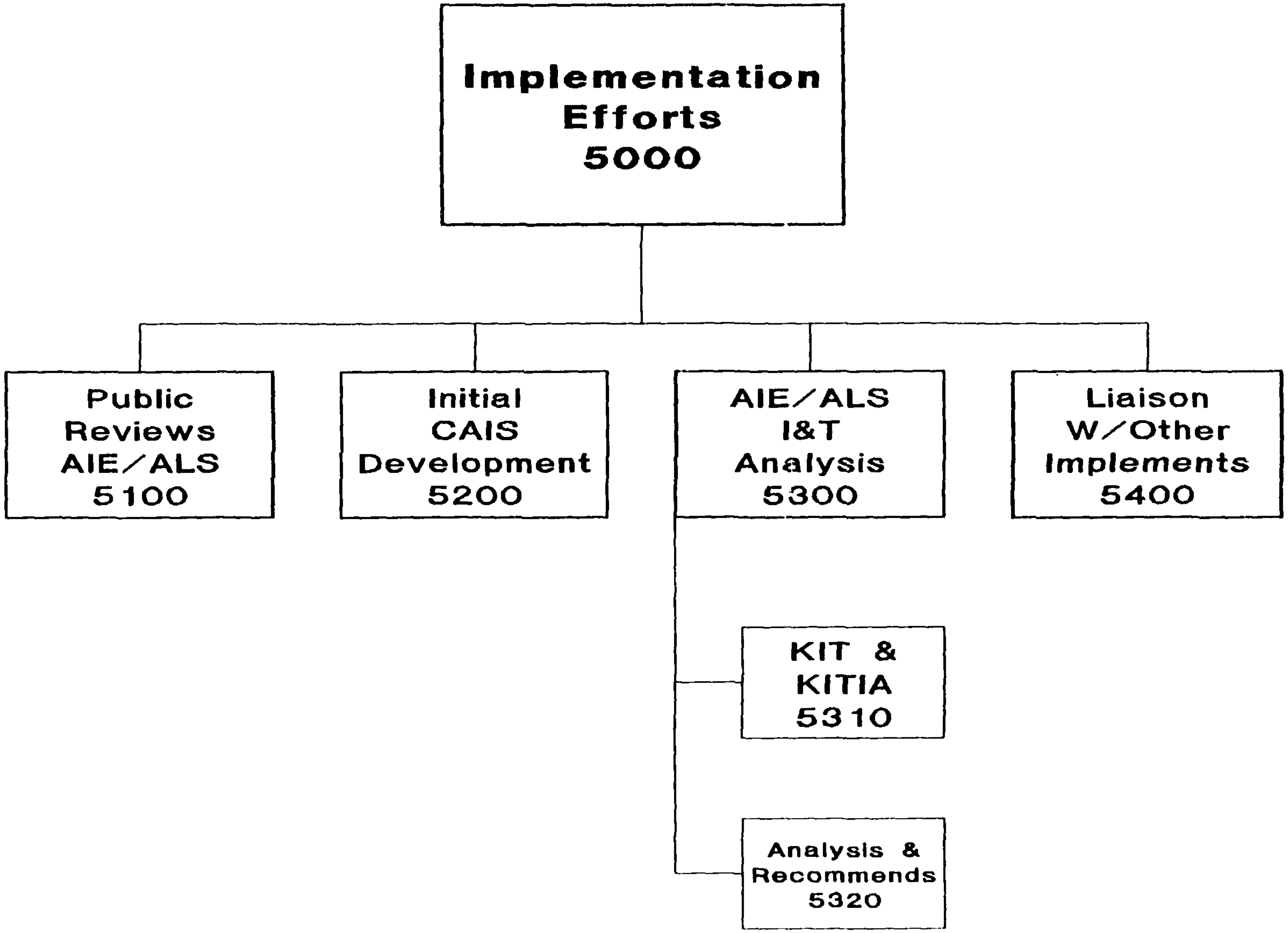


Figure 7. WBS Implementation Effort Overview

3.2 MAJOR APSE I&T DELIVERABLES

This section delineates the major deliverables of the APST I&T work.

3.2.1 APSE I&T Plan -

The APSE I&T Plan provides a detailed and organized approach to the accomplishment of the APSE I&T work. The plan reflects the management and technical approaches and the schedules for all activities. The plan is evolutionary and is updated annually to reflect changes in approach or schedule and to reflect accomplishments.

3.2.2 KIT Public Reports -

The KIT Public Reports are published every six months. Each one reflects the work that has been accomplished since the previous report, including deliverables, working group reports, position papers and meeting minutes.

3.2.3 Proposed Military Standard Common APSE Interface Set (MIL-STD-CAIS)

The proposed MIL-STD-CAIS is the specification of the interface set designed by the KIT/KITIA. This is the main objective of the APSE I&T effort. This document is accompanied by various supplemental ones, such as a Rationale, a Reader's Guide and an Implementor's Guide.

3.2.4 Requirements and Design Criteria (RAC) -

The Requirements and Design Criteria (RAC) document is intended to guide the development of CAIS Version 2. As a KIT deliverable, it is second only to the CAIS itself in importance. A companion RAC Rationale will also be generated by the KIT.

3.2.5 APSE I&T Guidelines and Conventions -

The Guidelines and Conventions covers those aspects of achieving I&T which are not directly addressed by the CAIS. They cover such things as using Ada to write transportable software and the style considerations which promote I&T. Two guides are under development: an Ada Transportability Guide and an APSE Interoperability Guide.

3.3 APSE I&T SCHEDULE

The schedule for the major APSE I&T events is given in Figure 8.

Figure 8 Schedule Summary

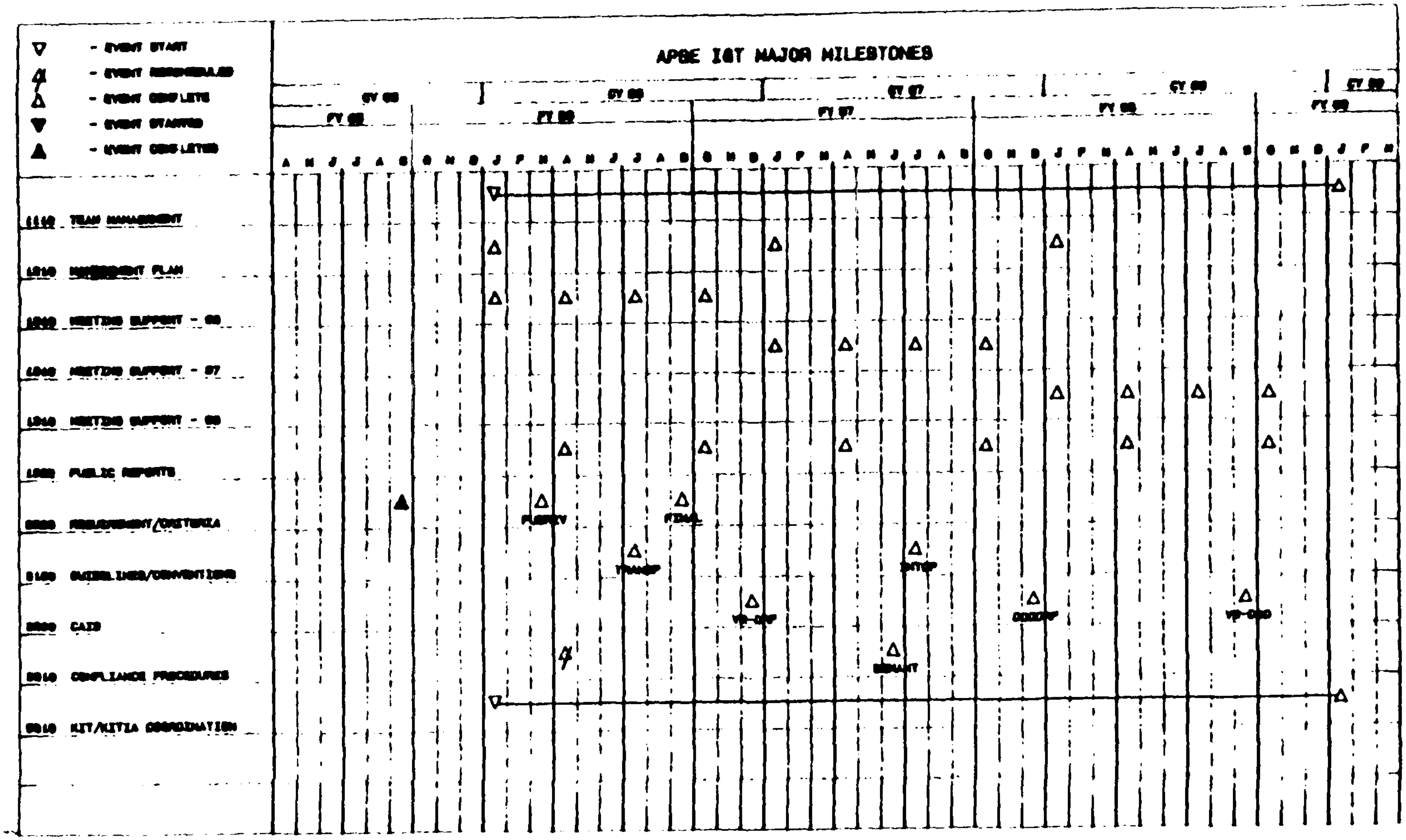


Figure 8 Schedule Summary

4.Ø PROVISIONS FOR SPECIAL NEEDS

This APSE I&T Plan emphasizes the development of requirements, conventions and standards. It is unusual in that it is written for a programming language support environment that is in the development state. At this point in development it is essential for the KIT/KITIA to provide an I&T forum and act as a focal point for the Ada community, APSE developers and the DoD. This will provide broad input to the KIT from which a complete, realistic set of I&T requirements, guidelines, conventions and standards will be developed that respond to ongoing APSE development and long term APSE needs.

Normally to achieve APSE I&T the APSE itself would be written in Ada. However, STONEMAN recognizes that "in cases where there is a large current investment in software projects, written originally in other languages", provisions and guidelines must be developed that account for cost effective transitions to Ada environments. In the development of APSE I&T requirements, conventions, and standards the KIT/KITIA will provide inputs that may supplement considerations for cost benefit analysis for different I&T implementation activities.

The STARS program has begun work in the last three years. The need for close coordination between the APSE I&T effort and various STARS projects is becoming increasingly apparent. However, it is not expected that this will have any noticable effect on APSE I&T schedules or objectives. It just helps to ensure that the work on the CAIS and other KIT efforts will be more closely related to the accomplishment of STARS objectives.

5.0 REFERENCE DOCUMENTS

Reference documents applicable to the APSE I&T effort include:

- Requirements For High Order Computer Programming Languages: STEELMAN, DoD, June 1978
- Requirements for Ada Programming Support Environments, STONEMAN, DoD, February 1980
- Kernel Ada Programming Support Environment (KAPSE) Interface Team: Public Report, Volume I, Naval Ocean Systems Center, Technical Document 509, 1 April 1982.
- Kernel Ada Programming Support Environment (KAPSE) Interface Team: Public Report, Volume II, Naval Ocean Systems Center, Technical Document 552, 28 October 1982.
- Kernel Ada Programming Support Environment (KAPSE) Interface Team: Public Report, Volume III, Naval Ocean Systems Center, Technical Document 552, 30 June 1983.
- Kernel Ada Programming Support Environment (KAPSE) Interface Team: Public Report, Volume IV, Naval Ocean Systems Center, Technical Document 552, 30 April 1984.
- Kernel Ada Programming Support Environment (KAPSE) Interface Team: Public Report, Volume V, Naval Ocean Systems Center, Technical Document 552, August 1985.
- Proposed Military Standard Common APSE Interface Set, Ada Joint Program Office, 31 January 1985.
- DoD Requirements and Design Criteria for the Common APSE Interface Set, KIT/KITIA, September 13, 1985.

APPENDIX A

GLOSSARY OF TERMS

Ada-JUG	Ada-JOVIAL Users Group
AIE	Ada Integrated Environment
AJPO	Ada Joint Program Office
ALS	Ada Language System
ALS/N	Ada Language System/Navy
ANSI	American National Standards Institute
APSE	Ada Programming Support Environment
DoD	Department of Defense
ECS	Embedded Computer System
FCDSSA	Fleet Combat Direction System Support Activity
GCS	Guidelines, Conventions and Standards
HOLWG	High Order Language Working Group
IOC	Initial Operational Capabilities
ISO	International Standards Organization
I&T	Interoperability and Transportability
JCL	Job Control Language
KAPSE	Kernel Ada Programming Support Environment
KIT	KAPSE Interface Team
KITIA	KAPSE Interface Team from Industry and Academia
MAPSE	Minimal Ada Programming Support Environment
NAC	Naval Avionics Center
NADC	Naval Air Development Center
NAVSEA	Naval Sea Systems Command
NAVSPAWAR	Naval Space and Warfare Command
NOSC	Naval Ocean Systems Center
NRL	Naval Research Laboratory
NSWC	Naval Surface Weapons Center
NUSC	Naval Underwater Systems Center
NWC	Naval Weapons Center
WBS	Work Breakdown Structure

APPENDIX B

WORK BREAKDOWN STRUCTURE TASK DESCRIPTIONS

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 1110

WBS ELEMENT TITLE: Team Management

PART OF WBS ELEMENT: 1100 - Project Management

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: NOSC Code 423 with KITIA Chairman support

TASK DESCRIPTION: Assemble original teams. Coordinate the solicitation and selection of new members. Organize team structure into working groups. Coordinate KIT and KITIA activities separately and together. Organize and coordinate all team meetings. Assign team and working group tasks and see to their completion. Plan and chair meetings. Cooordinate the raising and resolution of issues.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 1120

WBS ELEMENT TITLE: Presentations and Briefings

PART OF WBS ELEMENT: 1100 Project Management

DELIVERABLES/MILESTONES:	Project Review	May 1983	
	Senior Management Brief	Quarterly	
	SIGAda Conferences	October 1983	November 1985
		October 1984	November 1986

RESPONSIBILITY: NOSC Code 423 with AJPO support

TASK DESCRIPTION: Prepare slides and narration on team objectives, status, progress and plans. Present materials at project reviews, senior management briefings, SIGAda conferences, symposia, etc.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 1130

WBS ELEMENT TITLE: Coordination with Software for Adaptable Reliable Systems (STARS)

PART OF WBS ELEMENT: 1100 Project Management

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: NOSC Code 423 with AJPO

TASK DESCRIPTION: Attend STARS workshops. Cooperate with STARS personnel to assure proper incorporation of KIT/KITIA work into STARS plans. Participate in Software Engineering Environment (SEE) Team meetings and Software Engineering Environment Area Coordinating Team activities.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 1140

WBS ELEMENT TITLE: Coordination with Standards Community

PART OF WBS ELEMENT: 1100 Project Management

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: NOSC Code 423 with AJPO support

TASK DESCRIPTION: Keep standards community apprised of team activities and progress. Submit descriptions and reports as requested. Locate and track relevant standards activities.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 2

REVISION DATE: January 1986

WBS ELEMENT NR: 1150

WBS ELEMENT TITLE: Contracts

PART OF WBS ELEMENT: 1100 Project Management

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: NOSC Code 423

TASK DESCRIPTION: Initiate contracts and/or tasking necessary to achieve project objectives. At minimum, this includes contracts for tools, CAIS 2 design and general support. Monitor progress including reviews and examination of deliverables. Coordinate the incorporation of KIT/KITIA advice and comments into contract or tasks and of results of contracts into general KIT/KITIA work.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 1210

WBS ELEMENT TITLE: Management Plan

PART OF WBS ELEMENT: 1200 Planning

DELIVERABLES/MILESTONES: APSE I&T Management Plan

April 1983
January 1984
January 1985
January 1986
January 1987
January 1988

RESPONSIBILITY: NOSC Code 422

TASK DESCRIPTION: Plan activities as necessary to complete the APSE I&T project. Document all plans in the APSE I&T Management Plan. Update this plan once a year, or more often if radical changes occur.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 2

REVISION DATE: January 1985

WBS ELEMENT NR: 1220

WBS ELEMENT TITLE: Funding Allocation

PART OF WBS ELEMENT: 1200 Planning

DELIVERABLES/MILESTONES: Budget updates as required.

RESPONSIBILITY: NOSC Code 423

TASK DESCRIPTION: Establish budget for project activities. Secure funds as required. Manage the distribution and expenditure of funds by NOSC, contractors and other agencies. Update budget as necessary.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 2

REVISION DATE: January 1986

WBS ELEMENT NR: 1230

WBS ELEMENT TITLE: Strategy

PART OF WBS ELEMENT: 1200 Planning

DELIVERABLES/MILESTONES:	APSE I&T Implementation Strategy	May 1983
	Prototype paper	April 1985
	CAIS Tool Experiments	June 1986
	Security Investigations	May 1986

RESPONSIBILITY: NOSC Code 423

TASK DESCRIPTION: Establish, plan and document the strategy to be followed by KIT/KITIA in pursuit of APSE I&T objectives. Reflect this strategy in all plans, budgets and task assignments. Provide strategy working papers for particular subareas.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 1310

WBS ELEMENT TITLE: Meeting Support

PART OF WBS ELEMENT: 1300 Administrative Support

DELIVERABLES/MILESTONES: All support is required quarterly in conjunction with regular KIT/KITIA meetings. Other support is also required for special meetings and some working group activities.

RESPONSIBILITY: Support Contractors with NOSC Code 423.

TASK DESCRIPTION: Provide technical support required in planning, preparing for, conducting and reporting on APSE I&T meetings. Support includes, but is not limited to, the provision of agendas, discussion copies of papers, meeting arrangements, minutes and attendee lists.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1985

ORIGINATOR: NOSC

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 1320

WBS ELEMENT TITLE: Team Support

PART OF WBS ELEMENT: 1300 Administrative Support

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: Support contractor with NOSC Code 423.

TASK DESCRIPTION: Provide technical support required for maintenance, storage, updating and distribution of documents and data of the APSE I&T project. Support includes, but is not limited to, maintenance of address lists, document control, working paper preparation and MILNET directory administration, such as for KIT-INFORMATION and various comment directories.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 1331

WBS ELEMENT TITLE: Requirements, Guidelines, Conventions and Standards

---

PART OF WBS ELEMENT: 1330 Publications/1300 Administarative Support

---

DELIVERABLES/MILESTONES:

Requirements	December	1983
	Sept	1985
Guidelines/Conventions	June	1985
	July	1986
	July	1987
Standard	January	1985
	December	1987

RESPONSIBILITY: NOSC Code 423 with Support Contractor

TASK DESCRIPTION: Generate final versions of all named documents. Submit them to all appropriate publication processes. Provide for their distribution to the KIT/KITIA and to the public through NTIS.

NOTES: CAIS Version is available as NTIS report #A134 825.
Proposed MIL-STD-CAIS #

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 2

REVISION DATE: January 1985

WBS ELEMENT NR: 1332   WBS ELEMENT TITLE: Public Reports

PART OF WBS ELEMENT: 1330 Publications/1330 Administrative Support

---

DELIVERABLES/MILESTONES:

Public Report Vol.	III	April	1983
Public Report Vol.	IV	April	1984
Public Report Vol.	V	October	1984
Public Report Vol.	VI	April	1985
Public Report Vol.	VII	October	1985
Public Report Vol.	VIII	April	1986
Public Report Vol.	IX	October	1986
Public Report Vol.	X	April	1987

RESPONSIBILITY: NOSC Code 423 with Support Contractor

TASK DESCRIPTION: Generate publishable versions of all public reports. This includes determination and acquisition of contents, reformatting as necessary, organization, submission to publication process, distribution, notification of report availability and maintenance of the notification addressee list. Public distribution will be through NTIS.

NOTES: Volume I is NTIS #AD A1155 590
Volume II is NTIS #AD A123 136
Volume III is NTIS #AD A141 576
Volume IV AD A147 648
Volume V AD A160 355

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 1340

WBS ELEMENT TITLE: Correspondence

PART OF WBS ELEMENT: 1300 Administrative Support

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: All participants

TASK DESCRIPTION: Conduct communications as necessary, particularly using the MILNET. NOSC requirements in this element include the provision of terminals, ports and other required facilities in support of NOSC's other tasks.

NOTES:

WBS ELEMENT DESCRIPTION	ORIGINAL DATE: 30 April 1983
ORIGINATOR: NOSC	REVISION: 1
	REVISION DATE: January 1985

WBS ELEMENT NR: 1400 WBS ELEMENT TITLE: Configuration Management

PART OF WBS ELEMENT: 1000 APSE I&T Management

DELIVERABLES/MILESTONES:

Configuration Management Report	December 1983
Final Configuration Management Recommendations	January 1987

RESPONSIBILITY: NOSC Code 423 with Support Contractor

TASK DESCRIPTION: Plan for configuration management of tools and other products developed under this project. Perform configuration management during the project.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION:

REVISION DATE: January 1985

WBS ELEMENT NR: 2110

WBS ELEMENT TITLE: Relevant Research

PART OF WBS ELEMENT: 2100 Resource Reviews
Manchi's UNIX report December 1985
some of Herm's stuff, especially write PCTE

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: All participants

TASK DESCRIPTION: Review literature and documentation applicable to I&T requirements, guidelines, conventions and standards.

NOTES:

WBS ELEMENT DESCRIPTION ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC REVISION: 1

REVISION DATE: January 1986

WBS ELEMENT NR: 2120 WBS ELEMENT TITLE: Existing Standards

PART OF WBS ELEMENT: 2100 Resource Reviews

DELIVERABLES/MILESTONES: Continuous

CAIS Specification Coordination Report July 1984

RESPONSIBILITY: All participants

TASK DESCRIPTION: Locate and examine relevant standards. Use and/or incorporate relevant standards as found to be appropriate and applicable.

NOTES: As an example of this, the Operating System Command and Response Language (OSCRL) User Requirements, Functional Requirements and Design Criteria have been used as models for the APSE I&T Requirements and Criteria. The OSCRL documents were developed by X3H1.

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION:

REVISION DATE:

WBS ELEMENT NP: 2210

WBS ELEMENT TITLE: Definitions and Categories

PART OF WBS ELEMENT: 2200 Requirements Development

DELIVERABLES/MILESTONES: KAPSE Interface Worksheets December 1983

RESPONSIBILITY: All participants

TASK DESCRIPTION: Develop definitions of all relevant terms, particularly "interoperability" and "transportability". Develop categories of interfaces and KAPSE Interface Worksheets describing each of them.

NOTES: Completed

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 3200

WBS ELEMENT TITLE: Specification Development

PART OF WBS ELEMENT: 3000 APSE I&T Standards

DELIVERABLES/MILESTONES:

Common APSE Interface Set Specification Review	April 1984
CAIS Version 1 (Proposed MIL-STD)	January 1985
CAIS Version 2 Review Draft	December 1986
CAIS Version 2 (Proposed DoD-STD)	December 1987
CAIS Version 2 DoD-STD	September 1988

RESPONSIBILITY: NOSC Code 423 and all participants

TASK DESCRIPTION: Develop the set of interface specifications which will be recommended to the AJPO for standardization. Review and analyze these with respect to conformance with the requirements and criteria and to consistency, completeness and feasibility.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 3310 WBS ELEMENT TITLE: Compliance Procedures

---

PART OF WBS ELEMENT: 3300 Compliance

DELIVERABLES/MILESTONES:	Draft Compliance Procedures	June	1984
	Final Compliance Procedures	June	1987
	Formal Semantics paper	April	1986

RESPONSIBILITY: NOSC Code 423 with Support Contractor

TASK DESCRIPTION: Develop procedures for determining compliance of an APSE implementation with APSE I&T requirements, guidelines, conventions and standards. Apply these procedures experimentally to the I&T tools and the AIE and ALS. The results of this task will influence the form the standard specification will take. Coordinate with AJPO Evaluation and Validation (E&V) team.

NOTES: This compliance work will be conducted in close cooperation with the AJPO Evaluation and Validation team and will form the basis of the KIT/KITIA's recommendations to this team.

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 2

REVISION DATE: January 1986

WBS ELEMENT NR: 3320 WBS ELEMENT TITLE: Validation Recommendations

PART OF WBS ELEMENT: 3300 Compliance

DELIVERABLES/MILESTONES: Validation Recommendations January 1988

RESPONSIBILITY: NOSC Code 423 with Support Contractor

TASK DESCRIPTION: Review the results of the development and application of the Compliance Procedures (WBS 3310). Formulate recommendations for the AJPO and its Evaluation and Validation team.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 2220

WBS ELEMENT TITLE: Requirements and Design Criteria

PART OF WBS ELEMENT: 2220 Requirements Development

DELIVERABLES/MILESTONES:	Draft Requirements and Design Criteria	April	1984
	Revised Requirements and Design Criteria	Sept	1985
	Public Review	Mar	1986
	Final Requirements and Design Criteria	Sept	1986

RESPONSIBILITY: All participants

TASK DESCRIPTION: Develop functional requirements and interface design criteria for a set of interfaces which will achieve APSE I&T. Document and analyze these requirements and criteria. Analysis will be conducted through public review as well as team review and will determine completeness, consistency and feasibility.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 2

REVISION DATE: January 1985

WBS ELEMENT NR: 2300

WBS ELEMENT TITLE: Special Studies

PART OF WBS ELEMENT: 2000 APSE I&T Analysis

DELIVERABLES/MILESTONES: Workshops and reports as appropriate
Configuration Management workshop June 1983
Configuration Management report October 1983

RESPONSIBILITY: Various participants

TASK DESCRIPTION: Conduct technical analyses and studies as required. These special studies may include such topics as command languages, configuration management, STONEMAN revision and risks and cost benefits associated with various levels of I&T.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 1

REVISION DATE: January 1986

WBS ELEMENT NR: 3100

WBS ELEMENT TITLE: Guidelines and Conventions Development

PART OF WBS ELEMENT: 3000 APSE I&T Standards

DELIVERABLES/MILESTONES:

APSE I&T Guidelines and Conventions Review Draft	April	1984
APSE I&T Guidelines and Conventions Revision	October	1985
APSE I&T Guidelines and Conventions Final	January	1987
Transportability Guide	July	1986
Interoperability Guide	July	1987

RESPONSIBILITY: All participants

TASK DESCRIPTION: Develop guidelines and conventions for achieving I&T. These supplement and further explain the standard, covering those ideas and approaches that are believed to contribute to the achievement of I&T but which have not been included in the standard as yet or which cannot be achieved solely through common interfaces.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 3410

WBS ELEMENT TITLE: Experimental Implementation

PART OF WBS ELEMENT: 3400 Common APSE Interfce Set Analysis

DELIVERABLES/MILESTONES:

PA-APSE Implementation Report	June 1985
TI Report	1985
NOSC/Implementation Report	Feb 1986

RESPONSIBILITY: NOSC Code 423 with Support Contractors

TASK DESCRIPTION: Experimentally implement and exercise portions of the proposed common APSE interface set in order to investigate feasibility, completeness, etc. Report results as feedback to be incorporated in final common APSE interface set specification.

NOTES: Ref. Gould & MITRE reports/work   WP - 85W00537   01 Oct 1985

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 3

REVISION DATE: January 1986

WBS ELEMENT NR: 3420

WBS ELEMENT TITLE: Public Review

PART OF WBS ELEMENT: 3400 Common APSE Interface Set Analysis

DELIVERABLES/MILESTONES:	Review Version 1	September 1983	August 1984
	Review Version 1	November 1984	
	Review Version 2	Feb 1987	
	Review Version 2	July 1987	

RESPONSIBILITY: NOSC Code 423 and contractor

TASK DESCRIPTION: Present the proposed standard for widespread public review, including an open review meeting. Incorporate feedback in final document.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 2

REVISION DATE: January 1986

WBS ELEMENT NR: 3500

WBS ELEMENT TITLE: Standardization Process

PART OF WBS ELEMENT: 3000 APSE I&T Standards

DELIVERABLES/MILESTONES:

Initiate effort	May 1985
Standardize CAIS Version 1	Aug 1986
Standardize CAIS Version 2	Sept 1988

RESPONSIBILITY: NOSC Code 423 with AJPO and CAIS Control Board

TASK DESCRIPTION: Determine steps required to achieve standardization of the proposed interface set. Pursue standardization. Support the standization process.

NOTES: This activity alone among all these tasks may be expected to continue beyond the lifetime of the KIT/KITIA.

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 2

REVISION DATE: January 1985

WBS ELEMENT NR: 4100

WBS ELEMENT TITLE: Plans and Acquisition

PART OF WBS ELEMENT: 4000 APSE I&T Tools

DELIVERABLES/MILESTONES:

Plans	July	1982
Acquisition	October	1983

RESPONSIBILITY: NOSC Code 423

TASK DESCRIPTION: Identify the objectives, criteria and requirements to be used for the selection of three or more APSE tools. These tools will be used to further analyze interface requirements. Initiate acquisition of such tools.

NOTES: Completed

WBS ELEMENT DESCRIPTION	ORIGINAL DATE: 30 April 1983
ORIGINATOR: NOSC	REVISION: 3
	REVISION DATE: January 1986

WBS ELEMENT NR: 4200 WBS ELEMENT TITLE: Tool Development

PART OF WBS ELEMENT: 4000 APSE I&T Tools

DELIVERABLES/MILESTONES:

CMS Design	June	1983
AIM Implementation	June	1985
AIM Transport Experiment	July	1985

RESPONSIBILITY: Selected Contractors

TASK DESCRIPTION: Design, develop and test tools in a local environment. Provide insights into interface issues as they arise during development and integration.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 4300

WBS ELEMENT TITLE: Test and Analysis

PART OF WBS ELEMENT: 4000 APSE I&T Tools

DELIVERABLES/MILESTONES: Test Reports July 1985

RESPONSIBILITY: NOSC Code 423 with Support Contractor

TASK DESCRIPTION: Develop test applications and analyses for determining performance of APSE I&T tools in the AIE and ALS. Apply these to tools as they are completed.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 5100

WBS ELEMENT TITLE: Public Reviews of AIE and ALS

PART OF WBS ELEMENT: 5000 APSE I&T Coordination with Implementation Efforts

DELIVERABLES/MILESTONES: Public Review Reports July 1982 (ALS)
July 1983 (AIE)

RESPONSIBILITY: NOSC Code 423

TASK DESCRIPTION: Coordinate the establishment and notification of review teams. Determine documents or systems to be reviewed and arrange for distribution of copies to members of review teams. Receive all team review reports and correlate into report to AJPO and AIE/ALS sponsor.

NOTES: Completed

WBS ELEMENT DESCRIPTION

ORIGINAL DATE: 30 April 1983

ORIGINATOR: NOSC

REVISION: 2

REVISION DATE: January 1985

WBS ELEMENT NR: 5200

WBS ELEMENT TITLE: Initial Common APSE Interface Set Development

---

PART OF WBS ELEMENT: 5000 APSE I&T Coordination with Implementation Efforts

---

DELIVERABLES/MILESTONES: Initial CAIS Draft Report September 1983

RESPONSIBILITY: Selected participants with NOSC Code 423

TASK DESCRIPTION: Review AIE and ALS to determine a set of interfaces which is implementable in both of these systems. Develop a specification report documenting these interfaces. This task is to be accomplished with participation of AIE and ALS personnel.

NOTES: Completed

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 1

REVISION DATE: January 1985

WBS ELEMENT NR: 5310

WBS ELEMENT TITLE: KIT/KITIA Coordination

PART OF WBS ELEMENT: 5300 AIE/ALS I&T Analysis

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: NOSC Code 423

TASK DESCRIPTION: Provide channels of communication between KIT/KITIA members and government and contractor personnel involved in the AIE and ALS developments. Arrange for meetings and distribution of relevant documents. Provide feedback to AIE and ALS developers.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 2

REVISION DATE: January 1986

WBS ELEMENT NR: 5320

WBS ELEMENT TITLE: Analysis And Recommendations

PART OF WBS ELEMENT: 5300 AIE/ALS I&T Analysis

DELIVERABLES/MILESTONES:

ALS Analysis Report	May 1984
ALS & ALS/N Analysis Report	May 1988

RESPONSIBILITY: Various participants

TASK DESCRIPTION: Analyze AIE and ALS interfaces with respect to I&T. Provide recommendations for evaluation of each system to meet the interface set as it is put forward for standardization.

NOTES:

WBS ELEMENT DESCRIPTION

ORIGINATOR: NOSC

ORIGINAL DATE: 30 April 1983

REVISION: 1

REVISION DATE: January 1986

WBS ELEMENT NR: 5400

WBS ELEMENT TITLE: Liaison with Other Implementations

PART OF WBS ELEMENT: 5000 APSE I&T Coordination with Implementation Efforts

---

DELIVERABLES/MILESTONES: Continuous

RESPONSIBILITY: All participants

TASK DESCRIPTION: Maintain awareness of and contact with groups who are doing non-DoD APSE implementations. Solicit their inputs and provide information on KIT/KITIA activities. Examples of such groups are the UK, IABG in W. Germany, the EEC, ROLM, UC Irvine, Gould and MITRE.

NOTES:

CAIS 2
PROGRESS REPORT

KIT/KITIA MEETING
JULY 1986

SECURITY ISSUES
CAIS TYPING ISSUES
INPUT/OUTPUT ISSUES
PROTOTYPE

CAIS 2
PROGRESS REPORT

KIT/KITIA MEETING
JULY 1986

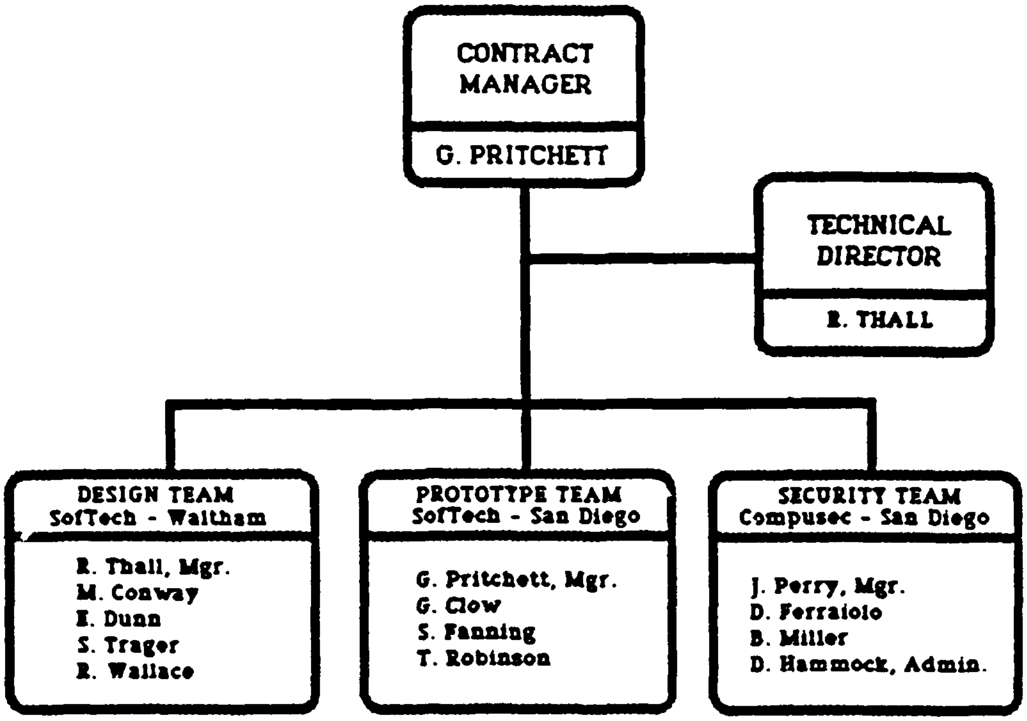
SECURITY ISSUES
CAIS TYPING ISSUES
INPUT/OUTPUT ISSUES
PROTOTYPE

Mark Conway
Ed Dunn
Dave Ferraiolo
Gary Pritchett
Rich Thall
Bob Wallace

AGENDA

1:15 - 1:30	INTRODUCTION New Project Members Bob Wallace Ed Dunn Sue Trager Dave Ferraioio Shawn Fanning	-- Rich Thall
1:30 - 2:15	SECURITY ISSUES	-- Dave Ferraiolo
2:15 - 2:30	CAIS TYPING ISSUES	-- Bob Wallace Mark Conway
2:30 - 2:45	BREAK (approx.)	
2:45 - 3:30	CAIS TYPING ISSUES (continued)	
3:30 - 4:15	INPUT/OUTPUT ISSUES	-- Ed Dunn
4:15 - 4:45	PROTOTYPE	-- Gary Pritchett
4:45 - 5:00	WRAP-UP	-- Rich Thall

THE CAIS 2 TEAM



CAIS 2 PROGRESS
JUL 86/3

NEW TEAM MEMBER

Bob Wallace - Principal Investigator

- BSEE - NORTHEASTERN UNIVERSITY
- 18 YEARS OF EXPERIENCE IN LARGE SCALE SYSTEM DEVELOPMENT
- CAIS RELEVANT EXPERIENCE:
  - SUPPORTING SEVERAL NAVY ORGANIZATIONS IN THE TRANSITION TO Ada* AND THE ALS THROUGH STUDIES, TRANSITION PLANNING, AND TEACHING.
  - ANALYSIS, DESIGN, AND PROTOTYPING OF A METHODOLOGY DRIVEN SYSTEMS ENGINEERING DEVELOPMENT ENVIRONMENT
  - DEVELOPMENT OF A DISTRIBUTED FAULT TOLERANT PROCESS CONTROL ENVIRONMENT USING Ada
  - MANAGER OF SOFTWARE TOOLS AND FACILITIES GROUP FOR A LARGE MILITARY CONTRACT
- AUTHOR OF "PRACTITIONER'S GUIDE TO Ada", McGRAW HILL, 1986.

* Ada is a registered trademark of the U. S. Government (AJPO)

CAIS 2 PROGRESS
JUL 86/4

NEW TEAM MEMBER

Ed Dunn - Systems Consultant

- BSCE - CASE WESTERN RESERVE UNIVERSITY
  MSCS - BROWN UNIVERSITY

- 9 YEARS OF EXPERIENCE IN REAL-TIME SOFTWARE DEVELOPMENT

- CAIS RELEVANT EXPERIENCE:

  - ANALYSIS, DESIGN, AND IMPLEMENTATION OF A VARIETY OF SYSTEM SOFTWARE FUNCTIONS AND OPERATING SYSTEMS EXTENSIONS FOR A DISTRIBUTED MULTI-COMPUTER/WORKSTATION REAL-TIME SYSTEM

  - DEVELOPED MODEL SOFTWARE STANDARDS AND PROCEDURES MANUAL PER MIL-STD-2167

  - WORKING KNOWLEDGE OF MULTIPLE PROGRAMMING LANGUAGES AND OPERATING SYSTEMS

  - ACTIVE MEMBER OF OPERATING SYSTEM USER GROUP FOR VARIOUS VENDORS

NEW TEAM MEMBER

Susan Trager - Systems Consultant

- BSCS/MATH - STATE UNIVERSITY OF NEW YORK AT ALBANY
  MSE - WANG INSTITUTE OF GRADUATE STUDIES

- 7 YEARS OF EXPERIENCE IN SOFTWARE ENGINEERING

- CAIS RELEVANT EXPERIENCE:

  - REQUIREMENTS ANALYSIS AND SPECIFICATION DEVELOPMENT OF THE STARS "SOFTWARE ARCHITECT'S WORKSTATION" - AN ELEMENT OF A METHODOLOGY DRIVEN SOFTWARE ENGINEERING SUPPORT ENVIRONMENT

  - WORKING KNOWLEDGE OF VARIOUS SOFTWARE ENGINEERING ENVIRONMENTS, OPERATING SYSTEMS AND PROGRAMMING LANGUAGES

NEW TEAM MEMBER

David Ferraiolo

- BSCS/MATH - STATE UNIVERSITY OF NEW YORK AT ALBANY
- 3.5 YEARS OF EXPERIENCE IN SOFTWARE ENGINEERING
- CAIS RELEVANT EXPERIENCE:
  - SOFTWARE DEVELOPMENT OF TRUSTED SYSTEMS APPLICATIONS AT THE NATIONAL COMPUTER SECURITY CENTER (NSA)
  - WORKING KNOWLEDGE OF VARIOUS OPERATING SYSTEMS AND PROGRAMMING LANGUAGES

CAIS 2 PROGRESS
JUL 86/7

NEW TEAM MEMBER

Shawn Fanning - Systems Consultant

- BACS - UNIVERSITY OF CALIFORNIA AT SAN DIEGO
- 10 YEARS OF EXPERIENCE IN THE DESIGN, DEVELOPMENT, AND MAINTENANCE OF COMPILERS, OPERATING SYSTEMS, AND SOFTWARE DEVELOPMENT ENVIRONMENTS
- CAIS RELEVANT EXPERIENCE:
  - REPRESENTED SOFTECH MICROSYSTEMS ON ANSI/IEEE Pascal STANDARDS COMMITTEE
  - DESIGN OF Ada RUNTIME SUPPORT LIBRARY FOR BARE VAX* TARGET
  - DESIGN AND IMPLEMENTATION OF A Pascal COMPILER AND Pascal BASED SOFTWARE DEVELOPMENT ENVIRONMENT

* - VAX is a registered trademark of Digital Equipment Corp

CAIS 2 PROGRESS
JUL 86/8

DELIVERY ORDER 3

- PRODUCE ISSUE REPORTS COVERING AT LEAST:
  - DATABASE TYPING
    DISTRIBUTION TAXONOMY
  - LOGICAL DEVICE DRIVERS
    FEATURES VS PERFORMANCE TRADEOFFS
    ACCESS CONTROL PERFORMANCE ISSUES
  - CAIS 2 ACCESS CONTROL
  - SECURITY VS DISTRIBUTION ISSUES
  - I/O PRIORITIES

- TOPIC IN THIS PRESENTATION

CAIS 2 PROGRESS
JUL 86/9

CAIS 2 SECURITY ISSUES

(SEE SEPARATE PACKET)

David Ferraiolo

CAIS 2 PROGRESS
JUL 86/10

CAIS 2 TYPING ISSUES

Bob Wallace
Mark Conway

OVERVIEW OF EMS TYPING

- FORMAL EMS TYPING IS NOT CONSIDERED IN CAIS 1
  - INFORMAL TYPING IS ACHIEVED WITH THE USE OF KINDS OF NODES/RELATIONS, NODE CONSTRAINTS, AND PREDEFINED OPERATIONS
- RAC REQUIREMENTS SPECIFY "DATABASE LIKE" TYPING FOR THE EMS
  - PROVIDE MEANS TO DESCRIBE DATA, DEFINING OPERATIONS AS LEGAL, ENFORCE DEFINITIONS, AND EXTEND DEFINITIONS
  - SEPARATE DESCRIPTION OF DATA FROM DATA INSTANCES
  - DEFINE NEW DATA DESCRIPTIONS THAT ARE DERIVED FROM EXISTING DATA DESCRIPTIONS
  - THREE CATEGORIES OF DATA TYPING REQUIRED
    ENTITY TYPES
    RELATIONSHIP TYPES
    ATTRIBUTE TYPES

SOME EMS TYPING ISSUES

- WHAT IS A TYPE DEFINITION?
  - ELEMENTS OF ENTITY TYPES
  - ELEMENTS OF RELATIONSHIP TYPES
  - ELEMENTS OF ATTRIBUTE TYPES
  - CONTENTS TYPING?
  - TRIGGER TYPING?
- HOW ARE NEW TYPES DEFINED?
  - SUPPORT FOR LATTICE DEFINITION
  - NAME RESOLUTION AND OTHER NAMING ISSUES
- HOW ARE TYPES IMPORTED AND EXPORTED?
- HOW AND WHEN TO CHANGE TYPE DEFINITIONS?

CAIS 2 PROGRESS
JUL 86/13

OUTLINE OF TYPING DESIGN STRATEGY

- ISSUE REPORTS IDENTIFIED A NEED TO HAVE A MECHANISM FOR EXPOSING ADDITIONAL DESIGN ISSUES
  - DO TRIGGERS HAVE TYPE DEFINITIONS?
  - NEED A FRAMEWORK IN WHICH TO TEST CURRENT THINKING AND DESIGN ALTERNATIVES
- BASIC TYPING DESIGN STRATEGY
  - DEVELOP AND DOCUMENT A "STRAWMAN" EMS DESIGN THAT INCLUDES TYPING
  - BASE "STRAWMAN" ON CAIS 1 WITH "DATABASE TYPING"
    - USE PCTE AND ADAPLEX AS MODELS
  - DOCUMENT "STRAWMAN" IN STORYBOARD FORMAT
  - ORGANIZE "STRAWMAN" ACCORDING TO CAIS 1 STANDARD OUTLINE
    - ADD SECTIONS WHERE REQUIRED

CAIS 2 PROGRESS
JUL 86/14

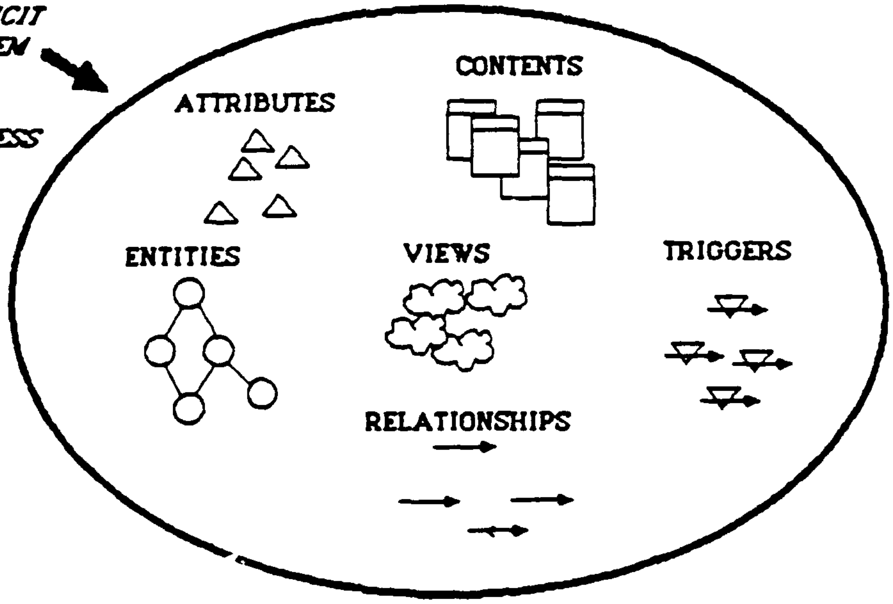
OUTLINE OF TYPING DESIGN STRATEGY (CONT.)

- TESTING THE DESIGN STRATEGY
  - VALIDATE COMPLIANCE WITH RAC AND RAC RATIONALE
  - TEST "STRAWMAN" FOR IMPLEMENTABILITY
  - USE AN EXTENDED RELATIONAL DBMS AS AN IMPLEMENTATION MODEL
  - REVISE "STRAWMAN" AS NEEDED
  - DOCUMENT ADDITIONAL ISSUES IN THE FORM OF ISSUE REPORTS

CAIS 2 PROGRESS
JUL 86/15

THE UNIVERSE OF EMS TYPING

CATEGORIES OF VIEWS (SUBSETS OF THE UNIVERSE):
IMPLICIT
SYSTEM
USER
TOOL
PROCESS



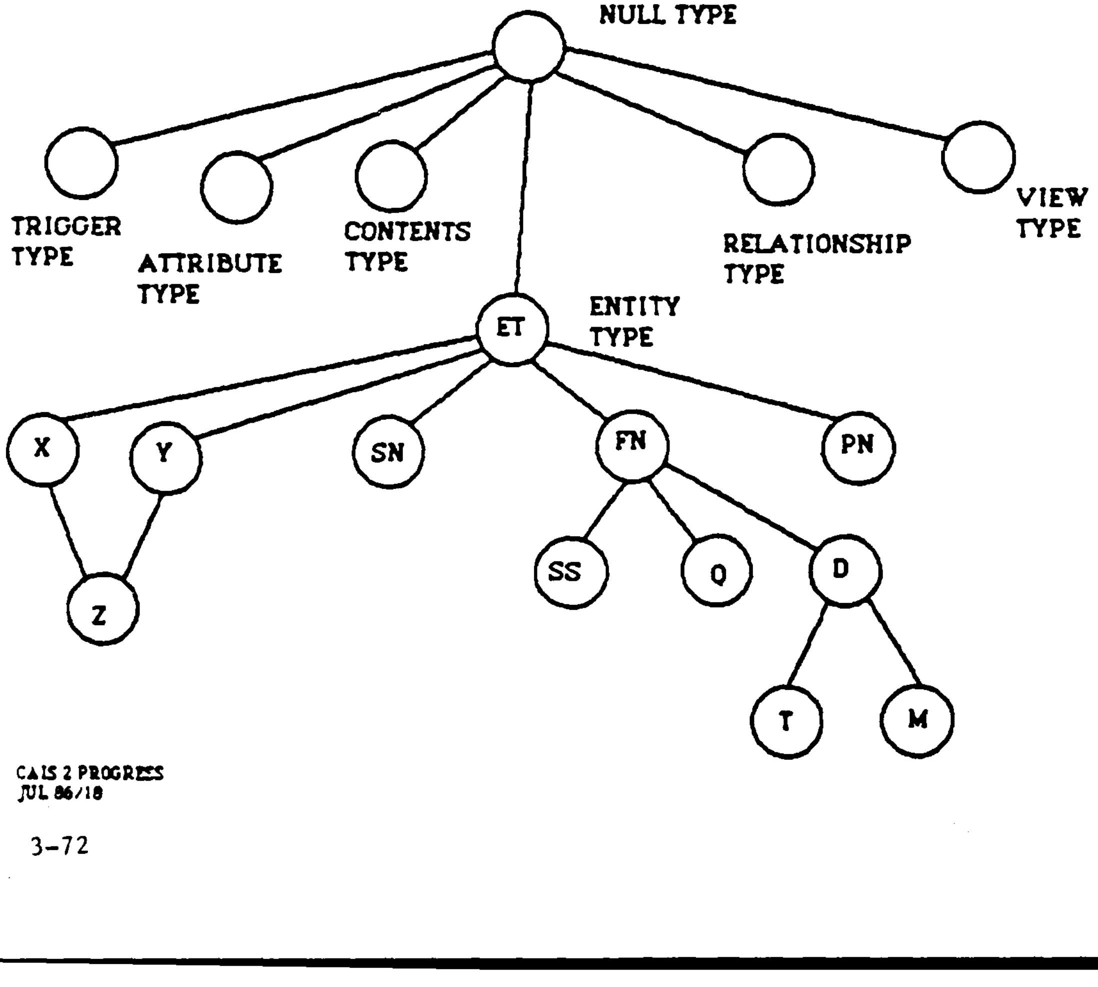
CAIS 2 PROGRESS
JUL 86/16

GENERAL FEATURES/ISSUES OF TYPING

- ALL TYPES ARE DEFINED FROM A SINGLE MASTER TYPE DEFINITION "THE NULL TYPE"
- CHARACTERISTICS OF TYPE DEFINITIONS VARY WITH TYPE CATEGORY
  - ENTITY TYPES FORM A LATTICE WITH INHERITANCE OF PROPERTIES
  - RELATIONSHIP AND ATTRIBUTE TYPES FORM FLAT STRUCTURES WITH NO INHERITANCE
- MULTIPLE VIEWS (SCHEMAS) ON THE TYPING UNIVERSE ARE ALLOWED
- A SINGLE CAIS DATA DEFINITION LANGUAGE WILL BE USED
  - OPERATIONS FOR USING THE DDL WILL BE PROVIDED AS CAIS SERVICES AND UTILITIES
  - CONTENTS TYPING WILL USE STANDARD Ada FACILITIES
- DOES TYPING IMPOSE RESTRICTIONS ON NAMING OF A TYPE OR INSTANCES OF A TYPE?

CAIS 2 PROGRESS
JUL 86/17

TYPING STRUCTURE



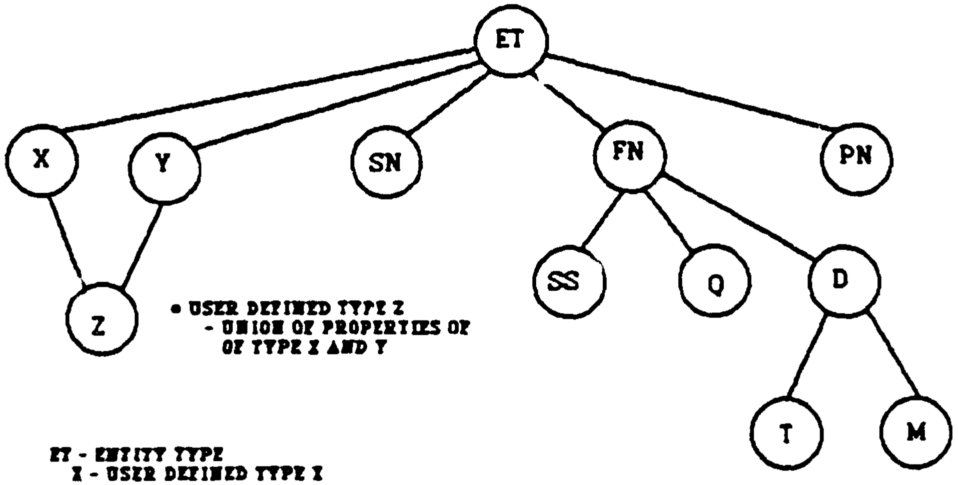
CAIS 2 PROGRESS
JUL 86/18

GENERAL FEATURES/ISSUES OF ENTITIES

- ENTITY TYPE CHARACTERISTICS
  - UNIQUE IDENTIFIER
  - NAME OF THE TYPE
  - MINIMAL SET OF ATTRIBUTE TYPES
    - MULTIPLICITY OF INSTANCES SPECIFIED AS A RANGE
  - SET OF RELATIONSHIP TYPES THAT START FROM THIS TYPE
    - MULTIPLICITY OF INSTANCES SPECIFIED AS A RANGE
  - ANCESTOR TYPES
- ENTITY INSTANCE CHARACTERISTICS
  - UNIQUE IDENTIFIER
  - TYPE
  - SET OF INSTANCE ATTRIBUTE TYPES (ONLY ASSOCIATED WITH AN INSTANCE)
- ENTITY CONTENTS ARE TYPED INDEPENDENT OF THE ENTITY TYPE
- HOW DO WE RESOLVE NAMING CONFLICTS ON MULTIPLE ATTRIBUTES OF THE SAME TYPE ATTACHED TO AN ENTITY?

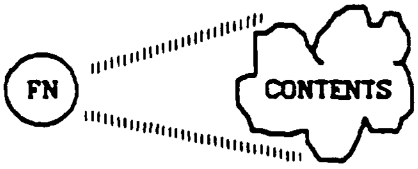
CAIS 2 PROGRESS
JUL 86/19

ENTITY TYPE LATTICE



- USER DEFINED TYPE Z
  - UNION OF PROPERTIES OF OF TYPE X AND Y

ET - ENTITY TYPE
X - USER DEFINED TYPE X
Y - USER DEFINED TYPE Y
Z - USER DEFINED TYPE Z
SN - STRUCTURAL NODE TYPE
FN - FILE NODE TYPE
SS - SECONDARY STORAGE
Q - QUEUE TYPE
D - DEVICE TYPE
T - TERMINAL TYPE
M - MAGTAPE TYPE
PN - PROCESS NODE TYPE



- CONTENTS ARE INDEPENDENTLY TYPED

CAIS 2 PROGRESS
JUL 86/20

GENERAL FEATURES/ISSUES OF PROCESSES

- PROCESS NODE TYPE CHARACTERISTICS

  - UNIQUE IDENTIFIER
  - NAME OF THE TYPE
  - CONTENTS
  - MINIMAL SET OF ATTRIBUTE TYPES

Status of Process	CURRENT_STATUS
Process (De)Activation	FINISH_TIME, START_TIME
Number of Open Handles	HANDLES_OPEN
Number of I/O Operations	IO_UNITS
Process Parameters	PARAMETER
Process Life	MACHINE_TIME
Process Result	RESULTS
Subject Class of Process	SUBJECT_CLASSIFICATION

  - SET OF RELATIONSHIP TYPE

	_ACCESS
To Role Node	ADOPTED_ROLE, ALLOW[redacted]
To File Node	CURRENT_ERROR, CURRENT_INPUT ..
To Any Node	CURRENT_NODE
To User Structural Node	CURRENT_USER, USER
To Top Device Node	DEVICE

  - ANCESTOR TYPE = ENTITY_TYPE

CAIS 2 PROGRESS
JUL 86/21

GENERAL FEATURES/ISSUES OF PROCESSES (CONT.)

- "SUBTYPES" OF PROCESS NODE TYPE

  - ROOT PROCESS NODE TYPE
  - SUBPROCESS NODE TYPE
    - SPAWNED TYPE
    - INVOKED TYPE

- TYPING OF PROCESSES WILL ALLOW US TO GENERALIZE THE OPERATIONS DEFINED IN THE CAIS 2 INTERFACE

  - ELIMINATES THE NEED FOR PROCESS UNIQUE ATTRIBUTE OPERATIONS

    APPEND_RESULTS, ... GET_RESULTS REPLACED BY GENERAL ATTRIBUTE OPERATIONS

  - REDUCES THE NUMBER OF PROCESS UNIQUE OPERATIONS

    REPLACED BY GENERALIZED ENTITY TYPE OPERATIONS

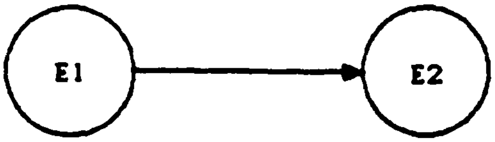
  - GENERALIZES SOME PROCESS OPERATIONS

    SUSPEND_PROCESS CAN ACT ACCORDING TO THE TYPE THAT IS PASSED TO IT

CAIS 2 PROGRESS
JUL 86/22

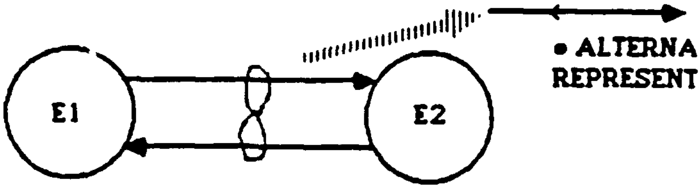
RELATIONSHIPS

SINGLE



- A UNIDIRECTIONAL LINK FROM E1 TO E2
  - A CAIS 1 RELATIONSHIP
  - A PCTE LINK

PAIRED



- ALTERNATE REPRESENTATION

- A BIDIRECTIONAL LINK BETWEEN E1 AND E2
  - A PCTE RELATIONSHIP

CAIS 2 PROGRESS
JUL 86/23

GENERAL FEATURES/ISSUES OF RELATIONSHIPS

- TWO FORMS OF RELATIONSHIPS
  - SINGLE (A CAIS 1 RELATIONSHIP)
  - PAIRED (A CAIS 1 PRIMARY/SECONDARY PAIR RELATIONSHIP)

- COMMON SET OF OPERATIONS ON ALL FORMS WITH SOME SPECIAL OPERATIONS ON PAIRED RELATIONSHIPS

- THERE IS NO HIERARCHY OF RELATIONSHIP TYPES

- SOME SINGLE AND PAIRED RELATIONSHIP TYPES WILL BE CAIS 2 DEFINED

- CREATING PAIRED RELATIONSHIP INSTANCES REQUIRES A TWO STEP PROCESS TO ENFORCE MAPPING

CAIS 2 PROGRESS
JUL 86/24

SINGLE RELATIONSHIP TYPE CHARACTERISTICS

- IDENTIFICATION
  - UNIQUE IDENTIFIER
  - NAME OF THE TYPE (CAIS 1 - RELATION NAME)
- PROPERTIES
  - MINIMAL SET OF ATTRIBUTE TYPES
    - MULTIPLICITY OF INSTANCES ALLOWED
  - SET OF KEY ATTRIBUTE TYPES
- SOURCE/TARGET
  - SET OF SOURCE/TARGET ENTITY TYPES
  - ARITY
  - STABILITY OF TARGET ENTITY TYPES (PCTE CONCEPT)
  - DEFAULT TARGET TYPE UPON CREATION OF RELATIONSHIP

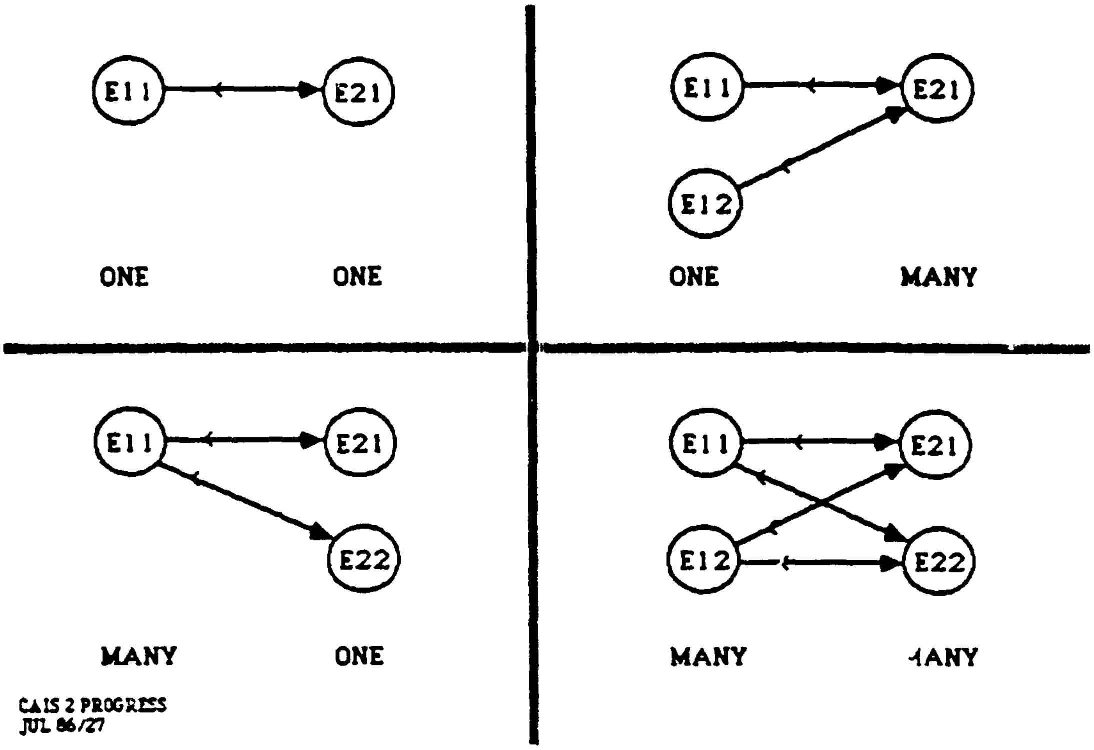
CAIS 2 PROGRESS
JUL 86/25

PAIRED RELATIONSHIP TYPE CHARACTERISTICS

- IDENTIFICATION
  - UNIQUE IDENTIFIER
  - NAME OF THE TYPE (IDENTIFIES THE FORWARD LINK)
  - NAME OF REVERSE LINK
    - DEFAULT NAMING WILL BE SUPPORTED
- PROPERTIES
  - MINIMAL SET OF ATTRIBUTE TYPES
    - MULTIPLICITY OF INSTANCES ALLOWED
  - SET OF KEY ATTRIBUTE TYPES
- SOURCE/TARGET
  - SET OF SOURCE/TARGET ENTITY TYPE PAIRS
  - ARITY (one-one, one-many, many-one, many-many)
  - STABILITY OF TARGET ENTITY TYPES (PCTE CONCEPT)
  - DEFAULT TARGET TYPE UPON CREATION OF RELATIONSHIP
- CAN WE DEFINE ALL RELATIONSHIPS TYPES AS PAIRED TYPES?

CAIS 2 PROGRESS
JUL 86/26

PAIRED RELATIONSHIPS MAPPING



CAIS 2 PROGRESS
JUL 86/27

GENERAL FEATURES/ISSUES OF ATTRIBUTES

- ATTRIBUTE TYPE CHARACTERISTICS
  - UNIQUE IDENTIFIER
  - NAME OF THE TYPE
  - VALUE TYPE (PREDEFINED IN CAIS 2)
    INTEGER
    DATE
    BOOLEAN
    DURATION
    LIST_TYPE
  - DEFAULT VALUE
  - RESTRICTIONS ON PREDEFINED OPERATIONS
- THERE iS NO HIERARCHY OF ATTRIBUTE TYPES
- APPLICATIONS OF ATTRIBUTES
  - PART OF TYPE DEFINITION FOR ENTITY AND RELATIONSHIP TYPES
  - ATTACHED TO AN INSTANCE OF AN ENTITY OR RELATIONSHIP
    (ALLOWS ADDITION OF ATTRIBUTES WITHOUT MODIFYING TYPES)

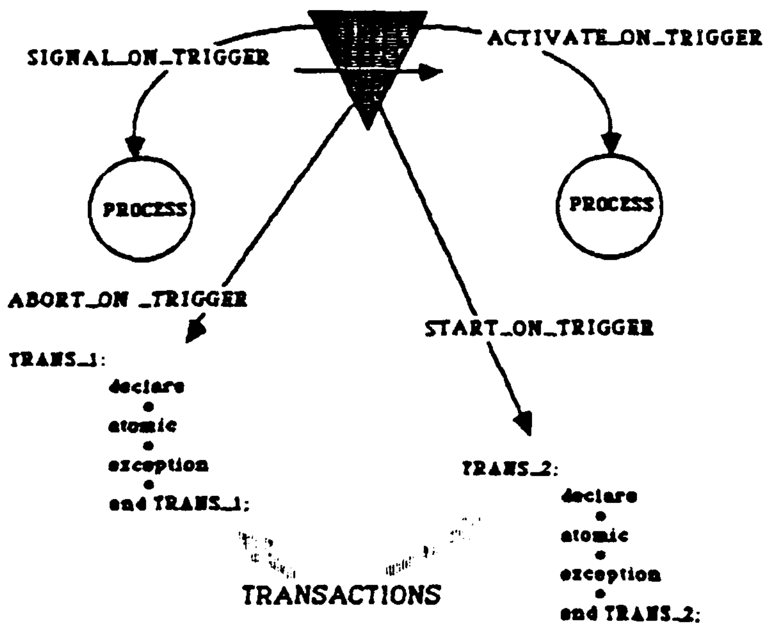
CAIS 2 PROGRESS
JUL 86/28

GENERAL FEATUkES/ISSUES OF ATTRIBUTES (CONT.)

- OPERATIONS ON ATTRIBUTE INSTANCES
  - CREATE AN ATTRIBUTE
    (WORKS FOR ENTITY AND RELATIONSHIP INSTANCES)
  - SET THE VALUE OF AN ATTRIBUTE
  - READ A VALUE OF AN ATTRIBUTE
  - APPEND A VALUE TO AN ATTRIBUTE (FOR LIST_TYPE VALUES)
  - DELETE AN ATTRIBUTE
- ISSUES
  - SHOULD RANGE CONSTRAINTS BE SPECIFIED IN ATTRIBUTE TYPE DEFINITIONS?
  - WHAT OTHER CAIS PREDEFINED VALUE TYPES SHOULD BE SUPPORTED?
  - SHOULD SUBTYPING OF ATTRIBUTES BE ALLOWED WITH SOME FORM OF INHERITANCE?

CAIS 2 PROGRESS
JUL 86/29

TRIGGERS



SIGNAL_ON_TRIGGER

ACTIVATE_ON_TRIGGER

PROCESS

PROCESS

ABORT_ON _TRIGGER

START_ON_TRIGGER

TRANS_1:
declare
•
atomic
•
exception
•
end TRANS_1;

TRANS_2:
declare
•
atomic
•
exception
•
end TRANS_2;

TRANSACTIONS

CAIS 2 PROGRESS
JUL 86/30

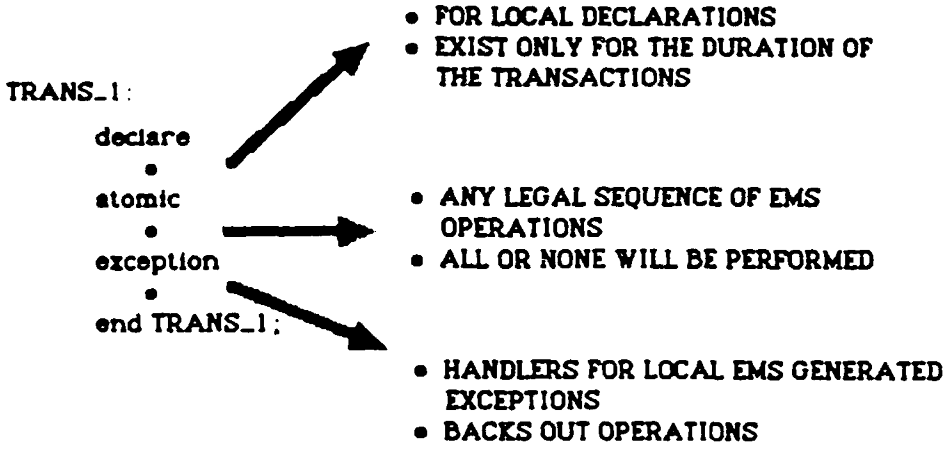
GENERAL FEATURES/ISSUES OF TRIGGER TYPES

- TRIGGERS ARE USED TO MONITOR THE DATABASE FOR THE OCCURRENCE OF CAIS AND USER DEFINED CONDITIONS
  - TRIGGER CONDITIONS CAN BE ANY LOGICAL COMBINATION OF:

    › DETECTED EVENTS IN EMS (CREATION OF A TYPE INSTANCE, ETC)
    › CHANGE IN ATTRIBUTE VALUE
    ANY CHANGE
    RELATIONAL OPERATION BASED
    MEMBERSHIP BASED

  - TRIGGERS ARE TYPED AND DEFINABLE USING THE DDL
  - TRIGGER INSTANCES ARE CREATED AND MANAGED WITHIN THE EMS
- TRIGGER INSTANCES CAN TRIGGER ONE OR MORE PROCESSES OR TRANSACTIONS
- WHEN ARE EVENTS IN THE EMS DETECTED?
  - PRE/POST AN OPERATION OR ONLY POST

TRANSACTIONS



```
TRANS_1:
    declare
       •
    atomic
       •
    exception
       •
    end TRANS_1;
```

- FOR LOCAL DECLARATIONS
- EXIST ONLY FOR THE DURATION OF THE TRANSACTIONS

- ANY LEGAL SEQUENCE OF EMS OPERATIONS
- ALL OR NONE WILL BE PERFORMED

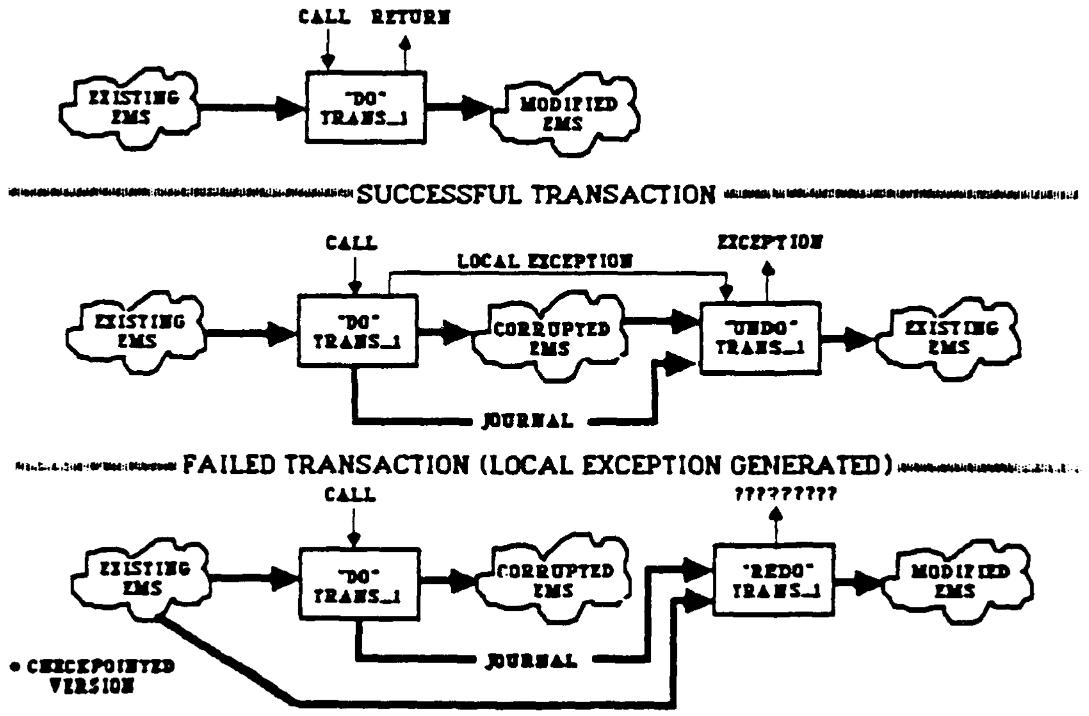
- HANDLERS FOR LOCAL EMS GENERATED EXCEPTIONS
- BACKS OUT OPERATIONS

GENERAL FEATURES/ISSUES OF TRANSACTIONS

- TRANSACTIONS ARE SEQUENCES OF EMS OPERATIONS DEFINED BY CAIS THAT ARE PERFORMED AS IF THEY WERE A SINGLE ATOMIC OPERATION
- TRANSACTIONS WILL BE DEFINED USING AN Ada LIKE LANGUAGE
  - THEY CAN BE CALLED USING A CAIS SERVICE
  - THEY CAN BE "CALLED" BY TRIGGERS
- HOW WILL RECOVERY FROM FAILURES BE TREATED IN CAIS?
  - WHEN DO WE CHECKPOINT?
  - HOW DO WE INFORM USERS OF TRANSACTION FAILURES THAT ARE CAUSED BY EQUIPMENT FAILURES?
- WHAT SERVICES AND UTILITIES MOST CAIS PROVIDE TO SUPPORT TRANSACTIONS?

CAIS 2 PROGRESS
JUL 86/33

BASIC PROCESS FOR TRANSACTIONS



SUCCESSFUL TRANSACTION

FAILED TRANSACTION (LOCAL EXCEPTION GENERATED)

FAILED TRANSACTION (EQUIPMENT FAILURE, ETC)

CAIS 2 PROGRESS
JUL 86/34

CAIS 2 INPUT/OUTPUT ISSUES

Ed Dunn

CAIS 2 PROGRESS
JUL 86/35

CAIS IO

- LOGICAL DEVICE DRIVERS - THEIR PLACE IN CAIS
- A CAIS-COMMON MECHANISM FOR HANDLING ASYNCHRONOUS EVENTS
- A UNIFYING MODEL OF IO AND INTERPROCESS COMMUNICATION

CAIS 2 PROGRESS
JULY 86 / 36

LOGICAL DEVICE DRIVERS

LOGICAL DEVICE DRIVERS

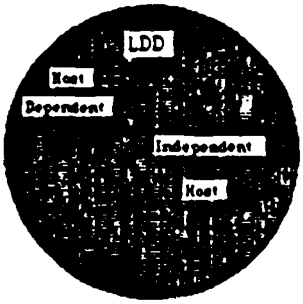
- DEFINITION

  A CODE FRAGMENT WHICH HIDES DEVICE DEPENDENCIES AMONG A CLASS OF SIMILAR DEVICES FROM CAIS TOOLS.

- A TOOL'S VIEW OF AN LDD

  - A CAIS PROCESS
  - A GROUP OF CAIS PROCESSES
  - A SPECIAL KIND OF DEVICE NODE
  - A PART OF CAIS OUTSIDE OF EMS MODEL
  - A GROUP OF PROCEDURAL INTERFACES

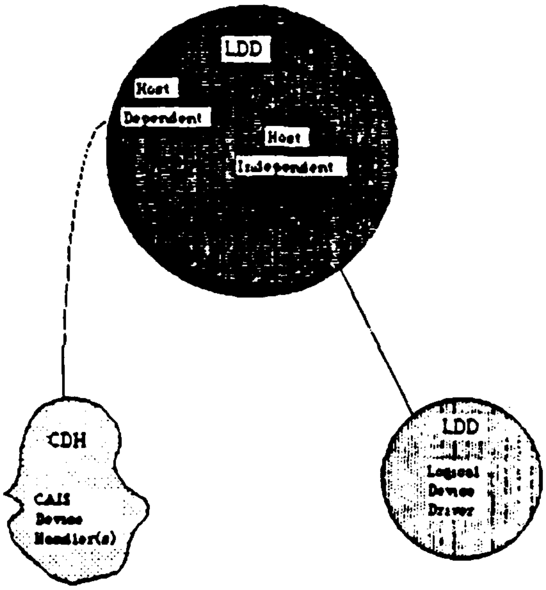
- COMPLETE PORTABILITY - IN THEORY
- IS IT REALISTIC?



CAIS 2 PROGRESS
JULY 88 / 39

SPLIT LDD

- TOOL PORTABILITY
- LDD PORTABILITY



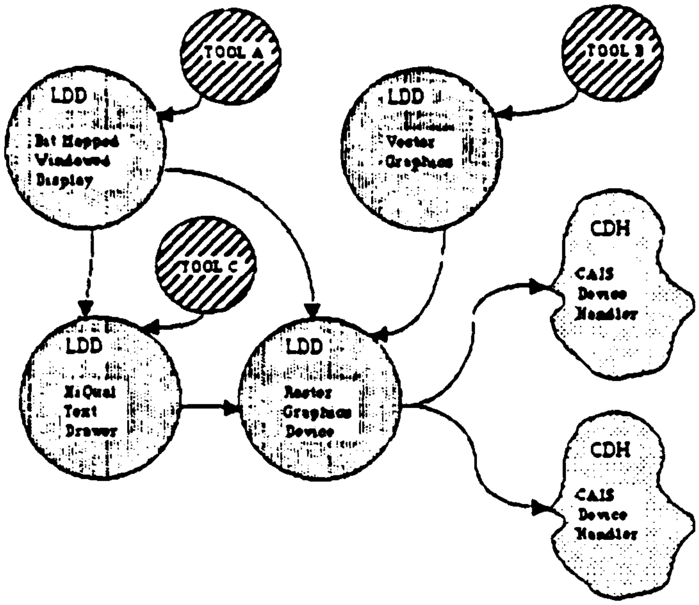
NOT PORTABLE BUT
STANDARD INTERFACE
TO LDD

PORTABLE WHERE
CDH IS AVAILABLE

CAIS 2 PROGRESS
JULY 88 / 40

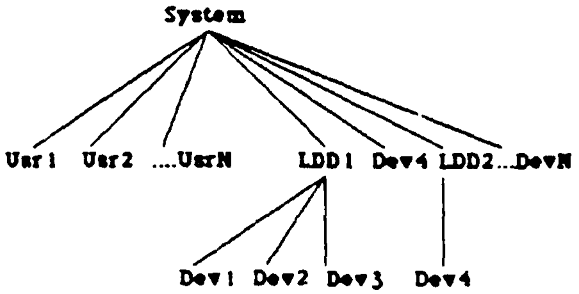
A GROUP OF CAIS PROCESSES

- ASSUMES CAIS DEVICE HANDLER AT LOWEST LEVEL



- DYNAMIC BINDING OF LDD PROCESSES TO TOOLS

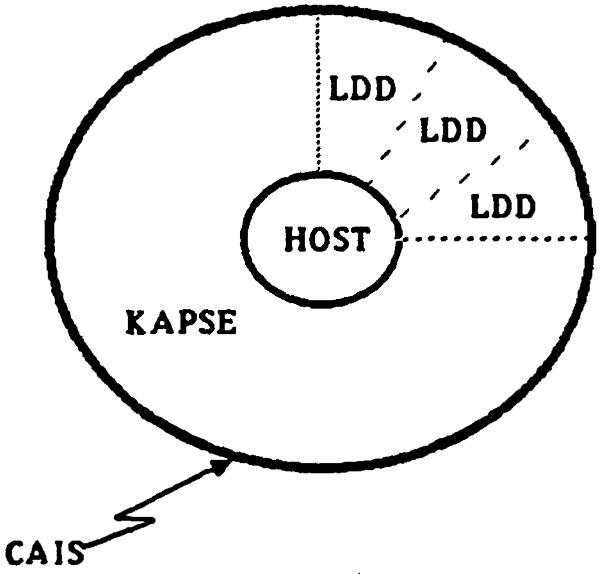
A SPECIAL KIND OF DEVICE NODE



- NOT MUTUALLY EXCLUSIVE WITH MOST OTHER CHOICES
- LDD'S SHOULD BE VISIBLE IN EMS MODEL

OUTSIDE EMS MODEL

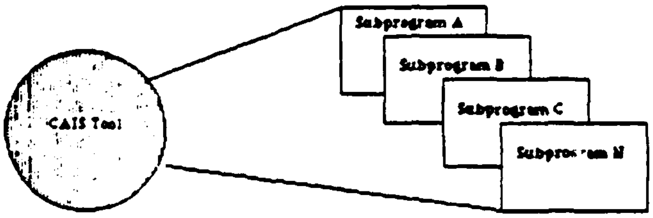
- CAIS INTERFACE DEFINED
- NO PORTABILITY OF LDD'S EXPECTED



CAIS 2 PROGRESS
JULY 86 / 43

A GROUP OF PROCEDURAL INTERFACES

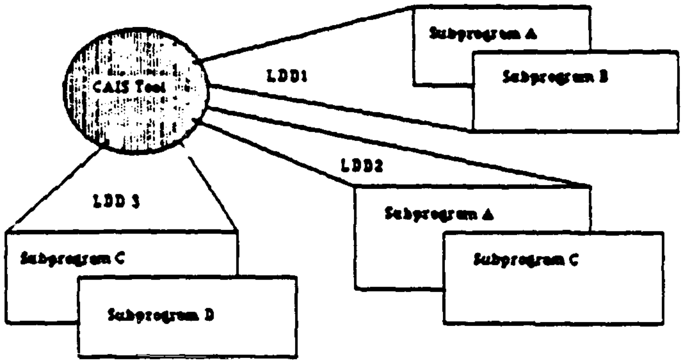
- SIMPLE LDD BUILT FROM LIBRARY



CAIS 2 PROGRESS
JULY 86 / 44

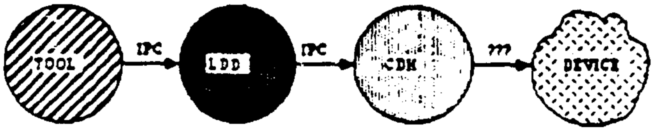
A GROUP OF PROCEDURAL INTERFACES

- MULITPLE LDD'S BUILT FROM COMMON LIBRARY



-SIMILAR TO PROCESS GROUP EXCEPT STATIC CONCEPT

RECOMMENDATIONS



- LDD'S SHOULD LOOK LIKE PROCESSES TO TOOLS

- AN LDD SHOULD NOT BE OUTSIDE THE NODE MODEL

  (A CAIS DEVICE HANDLER MAY BE)

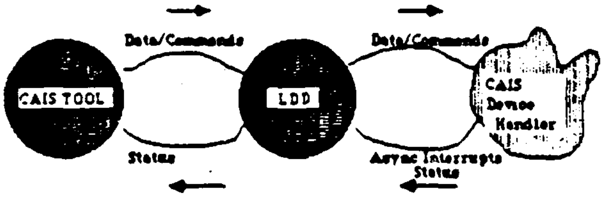
- PROCEDURAL INTERFACES ARE NOT LDD'S

  -COMMUNICATE WITH "NULL" LDD = CDH

A CAIS-COMMON METHOD FOR HANDLING ASYNCHRONOUS EVENTS

HANDLING ASYNCHRONOUS EVENTS

- LDD'S NEED EFFICIENT METHOD OF HANDLING DEVICE INTERRUPTS



- EXCEPTIONS AFFECT CONTROL FLOW
- DETERMINING STATUS AFTER EXCEPTION REQUIRES ADDITIONAL PROCEDURAL OVERHEAD
- HARDWARE INTERRUPTS NOT AVAILABLE TO LDD

RECOMMENDATION

- DEVELOP A CAIS-COMMON STRUCTURE FOR SOFTWARE INTERRUPTS AND STATUS INFORMATION

- LOOK LIKE STANDARD HARDWARE INTERRUPTS TO CAIS TOOLS

- COMPATIBLE WITH CAIS I / CHAPTER 14 10
  - AN EXTENSION, NOT A REPLACEMENT

ADA PSEUDO-CODE FRAGMENT OF ASYNCHRONOUS IO

```
OPEN(  CAIS parameters plus ..,SERVICE=>ASYNC,TRAPS=>SOFT),
READ_OK = TRUE,
NOT_EOF = TRUE,
INITIALIZE( MYDATA),
while NOT_EOF
loop
 if READ_OK then
   THIS_DATA = MYDATA,
   --
   --THIS_DATA is from initial value
   -- or completed READ of MYCATA
   --
   READ(STDINP,MYDATA),
   --
   --  Do Something useful with THIS_DATA, mydata is being read
   --
 else
   delay 0.5;
     ●
     ●
   --
   -- loop until read completes or gets bad status
   -- Trade-off between interrupt handler and
   -- this sub-program as to how much status-handling is
   -- done by each.
   --
     ●
     ●
 end if;
end loop;
```

CAIS 2 PROGRESS
JULY 86 / 50

ADA PSEUDO-CODE FRAGMENT OF INTERRUPT HANDLER FOR ASYNC IO

```
task STATUS_CHECK is
 entry STATUS_ARRIVED,
 for STATJS_ARRIVED use at STDINP ADDR,
end STATUS_CHECK;
task body STATUS_CHECK is
 begin
   loop
     accept STATUS_ARRIVED do
     -- do something "useful" with
     -- PORT_STDINP STATUS such as
     -- set flags for use by IO routine
     -- Example, READ_OK, EOF, etc
     -- and/or write errors to STDERR
     end STATUS_ARRIVED,
   end loop,
 end STATUS_CHECK,
```

CAIS 2 PROGRESS
JULY 86 / 51

ADA PSEUDO-CODE FRAGMENT OF SYSTEM TYPES TO SUPPORT IO

```
type SYSTEM_ADDR is
record
ADDR: VIRT_MACHINE_ADDR;
STATUS. STATUS_TYPE;
end record;

PORT array(1..23) of SYSTEM_ADDR
 :=(..initial values..); --Each VIRT_MACHINE_ADDR is unique per process


STDINP : SYSTEM_ADDR(21);
STDOUT . SYSTEM_ADDR(22);
STDERR : SYSTEM_ADDR(23);
```

NOTE. An alternative to the globally available status information is to pass it as a parameter into the interrupt handler, and make it accessible to tools via CAIS services.

PROPOSED VALID IO OPTIONS

1) Synchronous service with exceptions (normal chapter 14 / CAIS IO)

2) Synchronous service with software interrupts
   - -differs from normal in that you check your own status explicitly
   - -interrupts do not affect program flow like exceptions

3) Asynchronous service with software interrupts (example shown)

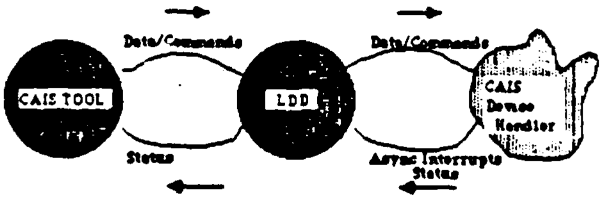
Asynchronous service with exceptions is NOT allowed as this would violate Ada definition of exceptions. (They could happen at any time, not just during CAIS procedure call.)

UNIFYING MODEL FOR IO AND IPC

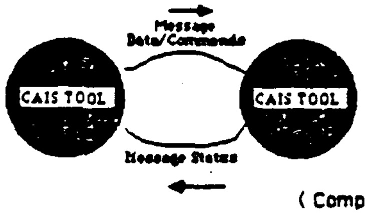
CAIS 2 PROGRESS
JULY 86 / 54

A COMMON IO / IPC MODEL

- REVIEW OF ONE METHOD OF CAIS IO



- PROPOSED METHOD OF .. .



- SENDMSG - NOWAIT
- SENDMSG - WAIT
- RECEIVEMSG - WAIT
- RECEIVEMSG -NOWAIT

(Companion path for reverse traffic)

- IO AND IPC USE SAME MODEL

CAIS 2 PROGRESS
JULY 86 / 55

ADVANTAGES OF COMMON IO / IPC MODEL

- DEVELOPMENT ADVANTAGES
  - DEBUGGING, TESTING, SIMULATION
  - LEARNING CAIS
- SIMPLIFICATION / UNIFICATION OF CAIS MODEL
  - REDUCES NUMBER OF INTERFACES
  - ELIMINATES REDUNDANT CONCEPTS (INVOCATION WITH PARAMETERS VS. IPC)
  - OFFERS CAIS-WIDE OPTION TO EXCEPTIONS FOR ERROR/STATUS REPORTING

DISADVANTAGES

- DIFFERS FROM CAIS 1 MODEL
- EASE OF REDIRECTION FROM IPC TO IO MAY CAUSE PROBLEMS WHEN DEVICE DEPENDENCIES ARE NOT PROPERLY MODELED BY PROCESS, OR VICE VERSA.
- PERFORMANCE DIFFERENCES AMONG POSSIBLE PROCESSES AND DEVICES ATTACHED TO A LOGICAL PORT MAY BE SIGNIFICANT, AND CAUSE "SURPRISES" TO UNPREPARED TOOLS

CAIS 2 PROTOTYPE

Gary Pritchett

PROTOTYPE GOALS

- PROVIDE PROTOTYPE SUPPORT FOR CAIS 2 DESIGNERS
- DEVELOP CAIS 2 PROTOTYPE
  - FUNCTIONALLY COMPLETE INCLUDING SECURITY, DISTRIBUTION, TYPING, AND DEMONSTRATION OF I & T
  - DEMONSTRATE IMPLEMENTABILITY
  - BASIS FOR TOOLS STUDY
  - PERFORMANCE NOT HIGH PRIORITY
- TUNE CAIS 2 PROTOTYPE
  - NEAR PRODUCTION LEVEL PERFORMANCE
  - MODEL FOR OTHER IMPLEMENTATIONS
  - BASIS FOR TOOL DEVELOPMENT AND USE

CURRENT ACTIVITIES

- DESIGN AND IMPLEMENT CAIS I NODE AND PROCESS MODELS
  - ALL NODE MODEL INTERFACES

  - INVOKE PROCESS, SPAWN PROCESS, AND CREATE JOB INTERFACES
  - NO OTHER PROCESS MANAGEMENT INTFRFACES

  - CREATE FILE NODE INTERFACE
  - LIMITED SUPPORT OF FILE CONTENTS
  - NO OTHER I/O INTERFACES

  - ENOUGH UTILITY INTERFACES TO DO ABOVE

DESIGN GOALS

- BASIS TO PRODUCF CAIS 2 DESIGN AND IMPLEMENTATION
- FLEXIBILITY FOR EASY EVOLUTION TO CAIS 2 IS A REQUIREMENT
- MODEL AFTER ALS FRAME AND CACHE IMPLEMENTATION
- SUPPORT MULTIPLE CONCURRENT USERS
- IMPLEMENT CAIS AS SHARED CODE IMAGES
- INSTRUMENT TO ALLOW DETAILED PERFORMANCE ANALYSIS
- PROVIDE A BASIS FOR THE ANALYSIS OF CAIS REHOSTABILITY

PROTOTYPE SCHEDULE

Draft Design	June 30, 1986 (completed)
Detailed Design Draft	August 29, 1986
Detailed Design Final	September 30, 1986
Node and Process Implementation	December 31, 1986
CAIS 2 Prototype	December 1987
Tuned Prototype	1988

CAIS 2 PROGRESS
JUL 86/62

WRAP-UP

Rich Thall

CAIS 2 PROGRESS
JUL 86/63

SOFTECH

COMMON APSE INTERFACE SET (CAIS)
VERSION 2
ISSUES REPORT

31 July 1986

SofTech, Inc.
16875 W. Bernardo Dr.
San Diego, CA 92127
(619)451-3350 TWX:710-324-6401

COMMON APSE INTERFACE SET (CAIS)
VERSION 2
ISSUES REPORT

31 July 1986

NOSC Contract No. N66001-86-D-0101
Delivery Order No. 0003
CDRL Item A002

SofTech Contract No. 1138

<u>Prepared for</u>

Naval Ocean Systems Center
San Diego, CA 92152-5000

<u>Prepared by</u>

SofTech, Inc.
460 Totten Pond Road
Waltham, MA 02254-9197

CONTENTS

ISSUE REPORT REVISION HISTORY

ISSUE 1.0 -- Preface

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	6/30		First Draft

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

CHAPTER 1

CHAPTER 1

PREFACE

In the process of designing the second version of the CAIS, the design team is investigating a number of the more important issues relating to the design. This evolving document is the record of the investigation of these issues. This is a foundation document which will be used as the basis of the design as well as the rationale. THESE ARE A COLLECTION OF WORKING PAPERS, authored by five or more contributors at varying levels of formality and completeness. Only a cosmetic level of uniformity has been imposed upon the authors' style. NO STATEMENTS MADE HERE SHOULD BE TAKEN AS CONCLUSIVE; these papers represent work in progress only.

The outline of this collection contains a number of items for which work has not yet progressed to the point where any useful written statement can be made. Such place-holders in the table-of-contents signify that we recognize the need for an issue statement on this subject.

In addition to these, there are still a number of important areas that should be considered for issue reports. These are:

a. built-in configuration management,

b. distribution as it relates to CAIS,

c. interoperability as an issue in its own right, although it is now rolled into some of the I/O discussion.

Chapters of this document constitute discussions of major subject areas. Chapter 2 is a collection of shorter subjects. Each chapter and each major section of Chapter 2, constitutes a report which may be revised and published separately. To control and document revisions, each chapter is preceded by a Revision History sheet describing the evolution of the document. Only technically significant changes are recorded in the revision history. Editorial alterations will be applied, as needed, without specific notice. Please note the applicable revision number when referring to any part of this document. In order to quickly gauge progress on the solution of technical issues, each chapter also has a summary of the issues treated. Each issue is marked OPEN or RESOLVED, to accurately portray the state of investigation.

ISSUE SUMMARY

ISSUE REPORT REVISION HISTORY

ISSUE 2.0 -- Short Issues

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	6/30		First Draft
1	7/31		Added multiple compiling systems issue

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

MAJOR ISSUE: Will there be fundamental changes to the NODE MODEL?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
2.0	Short Issues	
2.1	Proposed Streamlining of CAIS 1 Node Model	
2.1.1/0	OPEN	Should the distincition between primary and secondary relationships be eliminated? Should dangling relationships be eliminated? Should node existence semantics be changed? Should all relationships have implicit back links?
2.1.1/0	OPEN	Will we adopt the semantic data model? Will it be used just for typing, or will it have strucural implications, too?

ISSUE SUMMARY

MAJOR ISSUE: Which and how will other standards be incorporated into CAIS?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
2.2	Including Standards into the CAIS Framework	
2.2/0	OPEN	Alternatives are: by reference, by an Ada binding, orby full specification

ISSUE 2.3 -- Multiple Host Compiling Systems

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	7/30		First Draft

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

ISSUE SUMMARY

MAJOR ISSUE: How should CAIS implementations support multiple host compiling systems on heterogeneous distributed hosts?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
2.0 Short Issues		
2.3.1/0	OPEN	How do we achieve data format conversion to support interoperability?
2.3.3/0	RESOLVED	An "official" SINGLE HCS will be used to generate the CAIS implementation and all tools expected to execute on that host and use CAIS services, or produce data to be used by other programs that do use CAIS services.
2.3.3/0	RESOLVED	For a heterogeneous distributed CAIS implementation, a SINGLE HCS must be used to generate all CAIS kernels and tools expected to execute on any processor in the host and use CAIS services supplied by any processor in the host. The "official" HCS must also be used to generate any tools that will supply data to other programs that do use CAIS services.

CHAPTER 2

CHAPTER 2

SHORT ISSUES

This section discusses a number of short issues. Short does not imply minor. Understanding of some of these issues may require prior reading of other issues. Familiarity with Chapter 3 of this document will enhance the understanding of this section.

2.1 PROPOSED STREAMLINING OF CAIS 1 NODE MODEL

We currently plan to supply a typing mechanism which will allow the specification and enforcement of relationships which form a tree. Since CAIS 1 doesn't have this, primaries are needed. But, since CAIS 2 will have this, can primaries be eliminated?

2.1.1 Design Alternative

CAIS 1 requires that every entity have a primary relationship. An entity can always be reached by traversing the primary relationships. These are often DOT relationships, but do not have to be. The collection of all primary relationships forms at least one hierarchical representation of the database. Since a node can be a target of only one primary relationship, these can be used to find a partition of nodes without cycles in their relationships.

Secondary relationships can be created and deleted with no impact on the existence of an entity. In CAIS 1, relationships cannot be traversed in the reverse direction. Primary relationships have a reverse link, called the parent relationship. When a primary relationship is deleted, the entity is effectively deleted. This can leave dangling secondary relationships.

Primary relationships form both a known path to each node (if you know the name of the primary relationships) and an existence criterion.

The semantic that potentially leaves dangling secondary relationships has prompted many questions. One problem is the obvious concern for system integrity. What information may have been lost or capability lost due to the dangling relationships? This could cause serious holes in databases. When are the dangling relationships terminated? Immediately on node unobtainability, or by exception on the next attempted traversal? If the secondary relationship which was dangled is removed, does this change the type of the node from which it was removed? Is a such a change allowed? Does it render the node useless? How do you verify that it is permissible to dangle a relationship, when such relationships are one-way, implying the destination may not be aware of their attachment.

What other semantics could be used?

In view of the ability of relationship typing to enforce a tree structure, we propose a simplification of the fundamental node model semantics as follows:

a. The distinction between primary and secondary relationships be eliminated: relationships are relationships.

b. All relationships be traversable in the reverse direction (like PCTE).

c. A node becomes unobtainable when all of its incoming relationships are deleted.

This would require the ability to determine from a node which relationships were attached. This information would make navigation of the database a much simpler process, but requires a capability not currently in CAIS, that of "knowledge" of incoming relationships.

A hierarchical view of the database could be maintained by formulating the base entity type with a required parent-child relationship. It could also be handy to specify a standard name for this relationship, e.g., DOT, or PATH_ELEMENT. The tree structure would be enforced by the CAIS in response to the "arity" specification in the corresponding relationship type. In this scheme, there are no "dangling" relationships, an irksome feature of CAIS 1. Thus, the proposed powerful type mechanism is used in place of the primary-secondary mechanism in CAIS 1, helping to simplify the conceptual model. If this seems too roundabout, we can propose alternative specialized mechanisms for entity naming.

2.1.2 Recommended Design Resolution

An analysis of the CAIS 1 specification will be performed to determine the potential simplification resulting from eliminating the distinction between primary and secondary relationships. PCTE will be examined to determine its use of primary and secondary concepts.

2.1.3 Performance Impact of Recommended Design

The performance tradeoffs are unclear. Deleting entities with many incoming relationships may entail performance penalties as each relationship is retraversed to verify from the source that it is OK to delete it. This penalty may be needed to preserve database integrity. Other database integrity techniques may provide the same functionality without the need to incurr the overhead of maintaining the reverse links.

2.1.4 Compatibility of Recommended Design with CAIS 1

Any departure from CAIS 1 would cause interface specification changes, but not violate the goals of CAIS 1 as they are currently understood. These changes should result in a reduction of the number and complexity of interfaces. An overlying utility package could be used to supply CAIS 1 compatibility.

2.1.5 Security Issues for Recommended Design

None known.

2.1.6 Implementability of the Recommended Design

The modification of existence criteria to include all relationships instead of just the primary is viewed as an implementation simplification, not in its direct effects, but in the handling of special cases and side-effects of dangling relationships.

2.1.7 Other Design Issues

Is there some "hidden" use for primaries? Is an alternative partitioning method needed for backup, etc.?

2.2 INCLUDING STANDARDS INTO THE CAIS FRAMEWORK

Specific methods for incorporating another standard into CAIS 2 include:

a. By reference only.

b. By an Ada binding as GKS did.

c. By full specification in CAIS 2.

2.2.1 Design Alternatives

a. Reference standards.

b. Reference standards but include a "binding."

c. Fully specify in CAIS 2.

2.2.2 Recommended Design Resolution

Design alternative b will usually be preferable; however, the question must be addressed separately for each standard.

2.3 MULTIPLE HOST COMPILING SYSTEMS SUPPORTED BY ONE CAIS HOST

This issue involves the compiling systems used to generate Ada programs that execute on the CAIS host computer and use CAIS services. Can and should multiple host compiling systems be supported? What is required to support such a capability?

2.3.1 Definitions and Problem Scope

For this discussion, the term host compiling system (HCS) will refer only to the tools used to generate programs to execute on CAIS hosts. Tools used for generation of programs to run outside the CAIS compilation host are considered to be cross compilers for targets and are irrelevant to this discussion. Since cross compilation to other CAIS hosts, although not entirely irrelevant, is equivalent to target-only cross compilation, it is omitted from consideration in the following discussion.

A single HCS can include not only the Ada compiler, but also the run-time system (RTS), the program library manager, linker, loader, importer, and exporter. Debuggers, pretty printers, and other tools which require special knowledge of the RTS, intermediate languages, library structure, or code generation methods may also be included as part of an HCS. Such tools can embody many implementation and machine dependencies which affect the CAIS interfaces to tools as well as interfaces between CAIS implementations.

The term SINGLE HCS refers to an HCS which is specifically designed to follow compatible conventions for representing data objects and for protocols related to calling conventions, Ada task management, Ada exception handling, I/O, linking, and any other factors affecting the tool-CAIS interface. A SINGLE HCS may include many compilers, linkers, debuggers, etc.. However, by prearrangement, they must all subscribe to common representations and protocols. This term does not imply that all compilers in an HCS support Ada exclusively.

Multiple HCS's may arise for a number of reasons. Different HCS's can have different properties. One may be used for check-out and debugging, one used for high compilation speed, another for quality code exproduction.

Although the primary issue deals with multiple Ada compiling systems, this topic shares some concerns with the problem of supporting foreign languages, that is, with combining code written in multiple languages. The multiple HCS problem also embodies most of the problems of interoperability, that is, conversion of data representations. This issue also bears upon the question of how deep the CAIS standards should go. Finally, this issue impinges on RAC 5.5B, which states in part that "the implementation of the Ada RTS should be independent of the CAIS". This indicates that the CAIS designers must not place any requirements upon the RTS, further clouding the question of how multiple systems are to be coordinated.

2.3.2 Problems of Multiple HCS's

In order to accomodate multiple HCS's, it is necessary either to:

a. establish very detailed common conventions to be adopted by all HCS's, or

b. establish ways of converting between conventions.

Alternative (a) generally requires prearrangement by the HCS designers. Under the above HCS definition, this, in effect, is the same as creating a single HCS from the start. Common conventions must be established in the following areas:

a. data representations, including file representations, other I/O data streams, subprogram parameters, and shared data held in memory;

b. calling convention protocols, including parameter passing, stack management, return address handling, machine state storage and restoration, etc.; and

c. RTS protocols, including task dispatching, exception propagation, memory allocation, etc.

2.3.2.1 Examples of Representation Problems

Suppose there exist two HCS's, A and B, on one CAIS host. Further suppose that there is an Ada program P that creates and reads a file. Finally, suppose that program P is compiled with HCS A, call it P'A, and is used to create file F'A. The same program P can be compiled, without any change, using HCS B, giving P'B. Unless HCS A and HCS B subscribe to a common convention, there is no reason to expect that P'B can read file F'A. This incompatibility would result from such differences as differing integer sizes. record structure, and overall layout, etc. Analogous problems exist in any memory references for data.

2.3.2.2 Examples of Calling and Other Protocol Problems

Ada tools must be able to call and return CAIS services. In addition, during execution, Ada tools must coordinate with one another and with the provisions and expectations of the CAIS. Specific conventions for procedure calling and returning, task scheduling, stopping, signalling, and restarting, exception management, memory management, stack manipulation, etc., must all be mutually understood and agreed upon. Because the HCS incorporates the necessary RTS routines when linking, specific run-time mechanisms are all specific to one compiling system. The CAIS itself, developed with a particular HCS, incorporates such conventions implicitly and ubiquitously. Therefore, tools expecting to execute under CAIS must be compatible with CAIS, as compiled, in regard to these run-time concerns. The Ada standard does not establish any compatibility for these underlying mechanisms. In accordance with RAC 5.5B, as well as our own technical judgement, we do not believe that CAIS should specify or overly constrain designs at this level of detail, which would place additional requirements on this aspect of the compiling system.

2.3.3 Resolution

Conversion of I/O data stream representations, if well defined, is a viable alternative. However, conversion of other representations and protocols is a very dubious approach. Many of the conversions would be complex, or rendered unworkable by side-effects. Moreover, most of them occur so frequently that only very simple conversions would yield acceptable performance. It is for these reasons that the design team believes that requiring a single HCS is the superior technical approach. It is far simpler, avoids specifying a deep standard, and is more likely to yield acceptable results.

It is our understanding that the KIT has INFORMALLY agreed to the following guideline during discussion of this subject.

Given a homogeneous host, an "official" SINGLE HCS will be used to generate the CAIS implementation and all tools expected to execute on that host and use CAIS services, or produce data that will be used by other programs that do use CAIS services.

By extension, we assume that the following guideline would be acceptable for heterogeneous distributed systems.

For a heterogeneous distributed CAIS implementation, a SINGLE HCS must be used to generate all CAIS kernels and tools expected to execute on any processor in the host and use CAIS services supplied by any processor in the host. The "official" HCS must also be used to generate any tools that will supply data to other programs that do use CAIS services.

The above statement also covers homogeneous distributed implementations as a degenerate case.

These guidelines have two implications. First, in porting a tool, one must expect that recompilation using the "official" HCS will be necessary. Moreover, the "official" HCS must be robust enough to support compilation of all tools likely to be needed in a given situation. This has sweeping consequences for distributors of proprietary CAIS tools. Either they will have to distribute tools in source form, or some denatured (encoded) source form, or else they will have to take on the job of regenerating tools for each CAIS implementation, even on a single type of computer. The second implication is that the HCS implementors will be largely responsible for interoperability in heterogeneous systems. The CAIS is not a "magic bullet" which can meld diverse, separately developed HCS's into a working heterogeneous whole.

The only alternative to these guidelines would be to specify a very "deep" standard, constraining thousands of details of representations and protocols. We believe such a deep standard to be technically inadvisable, given the current state-of-the-art. The deepening of the standard should be left to implementors of heterogeneous distributed systems.

2.3.4 Notes on File Interoperability

NOTE

This section will be moved to a separate issue report on interoperability.

Ada tools should be able to use files created by other compiling systems. This is not directly possible because of differences in file layout. File interoperability pertains to both multiple HCS's and to the use of multiple languages. File interoperability, however, is frequently found to be a manageable probem.

One solution is to communicate desired files in an intermediate form such as a common ASCII file. A file to be transferred could be meaningfully read by a "transmitter program," a program developed under that file's original compiling system. This transmitter could then write a flat ASCII file having equivalent content but lacking the structural complexities of the input file. The flat ASCII file, in turn, could be read by a "receiver program" developed under the target compiling system. Once read, the ASCII data could be rewritten in a structure suitable for the target system.

Another approach to file interoperability assumes that all input/output is mediated by the RTS, and that all files have a layout specification in addition to their Ada specification. In this way, receiving files would be presented with sufficient information to interpret incoming files of arbitrary format and to construct equivalent files having specifically desired formats. Construction and interpretation of the layout specification would presumably be handled by the RTS, a situation possibly in conflict with the RAC.

ISSUE SUMMARY

ISSUE REPORT REVISION HISTORY

ISSUE 3.0 -- Typing in the CAIS Database

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	6/30		First Draft
1	7/31		Second draft -- no tech. changes

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

MAJOR ISSUE: Typing

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
3.0	Typing in the CAIS Database	
3.0	OPEN	What is the design of the CAIS typing mechanism?
3.1	Utility of Database Typing	
3.1/0		What are the expected benefits from typing?
3.1/0	OPEN	What performance degradation is acceptable? What complexity is acceptable for implementors? What complexity is acceptable by tool writers? (These can only be answered by demonstration.)
3.2	What is a Type Definition?	
3.2/0	OPEN	Are there to be separate type classes for: 1. Entity types 2. Relationship types 3. Attribute types 4. Triggers 5. Contents 6. Alternate views
3.2/0	RESOLVED	There will be no distinction between attribute types for entities and relationships. An attribute type can be attached to either a relationship or entity or both.
3.2/0	OPEN	Will we adopt the semantic data model? Will it be used just for typing, or will it have structural implications, too?
3.2/0	OPEN	What will the type lattice look like? What are the rules for inheriting and composing types of the various classes?
3.2/0	RESOLVED	All types will be derived from one root type, probably the null type.
3.2/0	OPEN	Will type names be used as instance names?
3.2.1	Entity Types	
3.2.1/0	OPEN	How will an entity type be defined?
3.2.1/0	RESOLVED	Will contents be just another attribute? No. Contents are special since there should be a one-to-one

		mapping between entities and contents. Multiple contents for one entity does not make sense, each content should be an entity in its own right. Directories have null contents, but files don't have multiple contents.
3.2.1/0	OPEN	How will contents be typed?
3.2.1/0	RESOLVED	A trigger type defines the conditions under which the trigger code is invoked. There may be many instantiations of triggers for each set of trigger conditions. It is not clear how it is decided which instantiation is invoked.
3.2.1/0	OPEN	Method of choosing which instantiation of a trigger type to invoke.
3.2.1/0	OPEN	The intersection of triggers and transactions is unclear. We would like to merge the concepts, if possible, to simplify the design.
3.2.2	Relationship Types	
3.2.2/0	OPEN	How will a relationship type be defined?
3.2.2/0	OPEN	Will relationship instances have names or will the type name be used? Will relationship instances have separate keys within each instance, or will relationship names and keys be the same concept?
3.2.2/0	OPEN	Will relationship type definitions be subsumed within entity type definitions under the semantic data model?
3.2.3	Attribute Types	
3.2.3/0	OPEN	How will an attribute type be defined?
3.2.3/0	OPEN	Will attribute instances have names or will the type name be used?
3.2.3/0	OPEN	Will attribute type definitions be subsumed within entity type definitions under the semantic data model?
3.2.3/0	OPEN	How will the representation of attribute values be typed, if at all?
3.2.3/0	RESOLVED	All attribute values will be byte strings. Further restriction to character strings is under strong consideration.
3.3	New Types	
3.3/0	RESOLVED	The CAIS standard will supply a fairly robust collection of standard type

definitions, a la Ada's Package Standard. All user-defined types will be derived from this standard set.

3.3.1 The Type Lattice and Defining New Types

3.3.1/0 RESOLVED The CAIS will define a standard DDL. Although the CAIS could just standardize the Ada call interfaces that register types with the CAIS implementation, this would fall short of a total solution. If the issue of DDLs is left to tool writers, a proliferation of DDLs will result, immensely complicating interoperability.

Export and import would then require porting DDL compilers or conversion of type definitions. This can all be avoided by standardizing the DDL.

3.3.1/0 OPEN Can the DDL definition wait until after the draft of CAIS 2?

3.3.2 Composing New Types from Existing Types

3.3.2/0 RESOLVED All types will be derived from a common root, probably the null type. The null type will be specified in CAIS operations where type checking is to be supressed, where safe.

3.3.2/0 RESOLVED It will be possible to build general tools which operate on all objects in the database and be insensitive to changes in the user-defined parts of the schema.

3.3.2/0 OPEN The semantic data model seems to have more consistent rules for derrivation of new types.

3.3.2/0 OPEN Reduction rules for determining type equivalence are important for importation of data and types.

ISSUE SUMMARY

3.3.3/0 Integration of New Types

3.3.3.1/1 Name Resolution

3.3.3.1/1 RESOLVED It will never be possible to totally automate importation of data and tools using type definitions defined remotely. However, tools can be built which assist by:

1. resolving types to a cannonical form (e.g., in terms of predefined types)
2. comparing type definitions, that is, identifying homonymy and synonymy
3. converting imported type definitions
4. converting types of imported data
5. converting type mapping of imported tools

3.3.3.2/1 Importing Foreign Types, Data, and Tools

3.3.3.3/1 Incompatibility Between Schemas

3.3.3.3/1 RESOLVED A DDL will be used to transport type definitions.

3.3.3.4/1 Incompatibility Between Data Areas

3.3.3.4/1 OPEN When moving parts of a database, how are severed relationships dealt with?

3.3.3.5/1 Recommended Design Resolution

3.3.3.5/1 RESOLVED At least part of the type importation problem can be simplified by assigning globally unique identifiers to type definitions themselves.

Then, the first level importation check can be done with the UIDs.

3.4/1 Changing Types

3.4.1/1 Changing Types

3.4.1/1 OPEN Define type modification. Define extension, and contraction of types.

3.4.1/1 OPEN What happens when an extant definition is changed? Is the DB converted immediately? Should types have revisions? Should the mechanism allow an open-endedness for easy extension at the cost of close control?

3.4.2/1 Deleting Types

3.4.2/1 OPEN When a type is deleted, what happens to DB instances? Do they become typeless, or become reduced to a predefined type?

3.4.3/1 Frequency of Type Changes

3.4.3/1 OPEN Type volatility is a major concern.

Should addition of types be allowed on-the-fly? How about extension or contraction of types?

3.4.3/1 RESOLVED On-the-fly type changing capability should be a configurable feature.

CHAPTER 3

TYPING IN THE CAIS DATABASE

A major requirement for CAIS 2 is to integrate database typing [1] into the entity-relationship model [2]; this model is the foundation for the CAIS 1 proposed standard [3]. This section examines the issues to be resolved in converting the CAIS from a partly-typed environment into a fully-typed one. Further, for each issue, basic design approaches will be offered.

The CAIS 1 node model supplies a solid basis upon which a typing mechanism can be imposed. First, the three basic classes of objects already exist: entities, relationships, and attributes. For CAIS 2, the objective is to partition each of these classes by typing the objects in them (RAC 4.2). Second, a good set of pre-defined entities, relationships, and attributes has already been defined by CAIS 1. For CAIS 2, the objective is to "type" these predefined objects and to link these types to form the primitive CAIS 2 lattice, upon which user types can be built (RAC 4.2B, last rule).

3.1 ISSUE: UTILITY OF DATABASE TYPING

Before examining other issues, the goals for typing in CAIS 2 must be unequivocally established. Since typing creates many problems, we must be sure that the costs of typing can be justified by the benefits. The RAC gives many mechanical requirements for typing, but does not establish how these mechanisms promote I and T. The major benefits expected from typing are:

a. To promote transportability. By supplying a means to control the name space for database objects used by tools. This will smooth the problem of tool importing.

b. To promote interoperability.

c. To promote database integrity.

d. To simplify tool writing by eliminating the need to check the nature of database objects.

e. To introduce redundancy which can be used for error detection.

Some of the major costs of typing are:

a. Reduced performance compared to typeless environments.

b. Increased implementation difficulties.

c. Increased semantic complexity for CAIS interfaces.

The balance between these advantages and disadvantages is delicate. Although this may degrade performance, its impact is hard to predict. There are many direct performance costs: database space required for type accounting, maintenance of the schema, retrieval of type information, and the setup and execution of type checking. But the indirect cost of contention for type information is likely to cause the most significant problems. The direct costs can be solved by increasing equipment capabilities; however, the cost of contention is a logical problem with a geometric growth. These performance problems will be accentuated when CAIS is compared with the performance of "typeless" environments.

Even though performance costs could negate any benefits of typing, we must still proceed vigorously with the design of a typing mechanism. Until we have considerable experience with operating CAIS implementations, we will be unable to convincingly establish where the complexity-integrity-performance balance lies. In fact, the balance is likely to be different for different CAIS users. Our best strategy is to design a mechanism that can adapt to the needs of the user. In particular, we do not wish to impose the more costly typing features on users unappreciative of the benefits (RAC 2.3).

The following issues are deferred until they can be examined with respect to a specific design proposal:

a. Upward compatibility from CAIS 1 to CAIS 2.

b. Security issues. The type schema and type side effects are a rich source of covert channels.

c. Implementability.

3.2 ISSUE: WHAT IS A TYPE DEFINITION?

Logically, there are two areas of data in the CAIS 2 database. One area is the data area; it contains instances of entities, relationships, and attributes. The other area is the data type area; it contains type definitions that describe the structure of the data area. The data type area will be called the schema. The schema contains the definitions of entity types, relationship types, and attribute types for every instance of entity, relationship, and attribute that can exist in the database. Note that the data type definitions could actually be represented as objects in the CAIS 2 database itself. This is a design question. (Curiously, this would require the definition of a "TYPE" entity type and associated attribute and relationship types. The method of initially entering the TYPE type in the schema is an interesting logical problem, but is not usually a practical problem.)

The main issue here is what properties a type is to assume. Are there separate type classes for entities, relationships, and attributes? Do these classes have a separate name space? Do types have unique identifiers? How is the type lattice formed? What are the inheritance rules?

3.2.1 Entity Types

An entity type can be defined using some or all of the following components:

a. Name identifier.

b. Unique identifier.

c. Set of mandatory relationship types possessed by entities of this type and the multiplicity of instances.

d. Set of optional relationship types possessed by entities of this type and the multiplicity of instances.

e. Set of mandatory attribute types possessed by entities of this type and the multiplicity of each.

f. Set of optional attribute types possessed by entities of this type and the multiplicity of each.

g. Set of operations allowed on entities of this type.

h. Ancestor types (rounds out the set of relationship types, attribute types, and operations).

i. Set of trigger types.

Note that trigger types extend the set of operations that can be performed on a certain entity type. That is, they may define a set of validation operations:

a. before or after creating a new type definition,

b. before or after changing a type definition, and

c. before or after deleting a type definition.

The effect of associating a trigger with the manipulation of a type definition is to define an implicit transaction. For example, if a type definition is changed, and there is an associated trigger, then the failure of the trigger operation causes the entire transaction to be backed out, leaving the original type definition intact.

3.2.2 Relationship Types

A relationship type can be defined using some or all of the following components:

a. Name identifier.

b. Unique identifier.

c. Set of names legal for relationships of this type.

d. Set of keys legal for relationships of this type.

e. Set of mandatory attribute types possessed by relationships of this type and the multiplicity of each.

f. Set of optional attribute types possessed by relationships of this type and the multiplicity of each.

g. Set of source-target entity type pairs legal for this relationship.

h. Type of mapping (one-to-one, one-to-many, etc.) legal for relationships of this type.

i. Set of operations legal for relationships of this type.

j. Stability of source entity imposed by relationships of this type (see below).

k. Stability of target entity imposed by relationships of this type (see below).

l. Ancestor types (rounds out the set of attribute types, source-target type pairs, and operations).

m. Set of trigger types

Stability is a property found in the ALS and PCTE [4]. In PCTE, relationships are created using two unidirectional links. Stability constrains changes allowed in an entity which is the destination (target) of certain links. For example, a link may be defined such that its destination entity cannot be deleted or updated as long as that relationship exists. A weaker level of stability would allow those entities to be updated, but not deleted.

3.2.3 Attribute Types

An attribute type can be defined using some or all of the following components:

a. Name identifier.

b. Unique identifier.

c. Set of names legal for attributes of this type.

d. Legal values for attributes of this type.

e. Set of operations legal for attributes of this type.

f. Ancestor types (rounds out the set of operations, may redefine the set of values).

g. Set of trigger types

3.3 ISSUE: NEW TYPES

The schema can be separated into two areas. One part is composed of the "system" (or predefined) type definitions. We will call these CAIS defined types, since they will appear in the standard. The other part of the schema is composed of user-defined types. User-defined types are either extensions or modifications of the system types.

3.3.1 The Type Lattice and Defining New Types

Typically, database systems define a Data Definition Language (DDL) that is an interface to the data type definition facilities. With the DDL is a tool that converts DDL statements into an underlying representation used to communicate with the data type facilities. In order to allow for interchange of type definitions, CAIS 2 should define a specific DDL. (Technically, this is a violation of RAC 3.1A which requires CAIS syntax to be defined as Ada Package Specifications.)

An alternative to using a DDL is to employ the semantics of the Ada language. That is, a database type could be represented using an Ada abstract data type. Here, the expressive power of Ada is good for defining complex types, but there are some problems with enforcement of data representations. A proposal from Robert Munck of Mitre explaining this approach is included in Appendix A.

3.3.2 Composing New Types from Existing Types

Our current thinking employs the concept of a lattice of types. The "root" of the lattice would consist of the null type. The next level would have three derivative types: one for all Entity (E) Types, one for all Relationship (R) Types, and one for all Attribute (A) Types. All other types would be derivatives of these. The null type could not be used for the creation of an object, but could be specified as the type to use when no type checking is desired. (Not all tools or users may be given this ability.) Depending upon the rules for deriving new types, the type lattice may, in fact, be a tree. Another issue to resolve is what level of type control is to be supplied for attribute value representations and how such representations should fit into the type lattice.

```
                   Type Lattice

                    Null_type
                     /  |
                    /   |
                   /    |
                  /     |
              E_Type  R_Type  A_Type
              //|    /|      /|      CAIS DEFINED TYPES
               / |    ...     ...
             /  |
            /   |
           /    |
      Process Device File
        /|     /|   /|
        ...    ...   ...
     ----------------------------------

                                       USER DEFINED TYPES
```

It is important to be able to compare types for structural equivalence. It will be important for a user to know when he is creating a type that is structurally equivalent to one or more extant types. He can then decide if the new type is really the same as an existing type. To do this, the CAIS should provide a mechanism whereby user types can be reduced to a composition of CAIS-defined types. A tree-structured lattice supplies an obvious way to perform this reduction. Reduction rules for a more general lattice are not obvious.

3.3.3 Integration of New Types

There are two main issues when a new type is added to the schema. First, the CAIS must resolve any naming conflicts. Second, it must preserve the integrity of the type lattice.

3.3.4 Name Resolution

There are two kinds of naming conflicts. First, if a type to be added has the same name but different structure as an existing type, then the two types probably represent different concepts. This conflict (called homonymy) can be resolved by asking the user for a different name. (This is more complicated when the problem arises during data or tool importing.)

The other naming conflict is much harder. In this case, the type to be added is structurally equivalent to one or more existing types with different names (called synonymy). If this conflict exists, then the type lattice is redundant, and there is unnecessary duplication. Only the user can determine if the intended new type is the same as any of the existing types, i.e., it has the same purpose as an existing entity, attribute, or relationship type. However, to avoid proliferation of types, the user should be made aware of all synonyms whenever a new type is installed.

3.4 ISSUE: IMPORTING FOREIGN TYPES, DATA, AND TOOLS

A major goal of database typing is to facilitate the integration of new tools and new data with an extant database. It must be possible to take a data subset from one database and incorporate it into another database, similarly for the associated data types. The problem would be simplified if only instances of CAIS-defined types could be transported. Even though such a restriction is obviously unacceptable, it does highlight the usefulness of being able to reduce all user-defined types to a composition of their CAIS-defined types. (This would be used as an equivalence check; we are not proposing to prevent the interchange of user-defined types.)

Successful importing of types and data instances requires that any incompatibility between the source and target schemas and between the source and target data must be identified and resolved. Incompatibility results from synonymy and homonymy of types, instances of entities, instances of attributes, instances of relationships, and data instances unsupported by type definitions.

3.4.1 Incompatibility Between Schemas

The source schema might contain a type that does not exist in the target schema. The foreign type can be entered in the target schema by either reducing the new type to CAIS-defined types or supplying all intermediate type definitions upon which the new type depends. The potential for incompatibility also exists for every intermediate type imported. Imported tools may use type definitions, for which no instances actually exist in the database. Even if the entity could be stored, the naming conflicts analysis would have to be performed. Every importing becomes a two-step process: first, to resolve type conflicts, then to resolve conflicts in the imported data and tools.

In order to exchange type information, a standard language (DDL) and representation will have to be adopted. This is another interoperability problem that will require the use of the Common External Form. In [5], the exported types and instances are each in a physical format specified by the International Standard ISO 8211, "Specification for a Data Descriptive File for Information Interchange."

3.4.2 Incompatibility Between Data Areas

The same name resolution problems exist with instances of named database objects as exist with type names.

In a highly interconnected database such as the E-R model encourages, exporting of a data subset can involve severing many relationships. We expect that the proper restoration of these relationships will prove to be a difficult problem, a problem which is complicated by the need to restore relationships with the correct entity types and mapping. It may well prove that an organized, clustered arrangement of entities is the best solution for this issue.

3.4.3 Recommended Design Resolution

We are currently exploring the utility of universally unique identifiers (UIDs) as a means of quickly and automatically resolving some of the name resolution questions, for both types and data instances. UIDs are used by the ALS, PCTE (definition-ID), and are required by the RAC. The RAC, however, does not explicitly connect types and UIDs. UIDs would be imported as part of type definitions. Types could be matched by UID even though they may have been renamed. The advantage of a UID is that it is an unambiguous identifier. If the UIDs match, the types match. Homonymy of the mnemonic names is still an issue, however.

3.5 ISSUE: CHANGING TYPES

Part of this issue is to assess the impact of changing a type on the fly. The greater part of this issue is to define a good set of rules for creating types, changing types, and deleting types. Existing system specifications are con idered.

3.5.1 Changing Types

This is one of the more complex typing problems. The first issue to examine is what constitutes a type modification. This issue should be resolved once the definitions of an entity type, a relationship type, and an attribute type are established. The second issue to examine is the extension and contraction of types and the effect on the type lattice. For example, what happens to instances of a type when some of the type's components are deleted? Is the use of type revisions a good idea?

Clearly, modifying types is dangerous because it affects the operation of tools. The alternative to modifying a type is to create a new one based on the original, leaving the original intact. On the other hand, this alternative could easily lead to a plethora of types. This alternative could be more attractive if type revisions were used. Extant instances would refer to specific type revisions, while new instances would use the latest revision. It is not clear, however, how tools would deal with type revisions.

3.5.2 Deleting Types

Examine how a type's instances are affected when a type definition is deleted. If a type definition is deleted, should the instances of that type be deleted or just become "typeless"? Can a type definition be deleted if there are any instances of that type in the database?

3.5.3 Frequency of Type Changes

The frequency of type changes may be so high that performance would be terrible. It should be possible to configure CAIS so that type changes would not be allowed.

3.6 ISSUE: MAPPING OF RELATIONSHIPS

Functional Mapping or Relational Mapping? Arity Control as in PCTE?

3.7 ISSUE: NAME SPACE CONTROL

Name spaces of entities, relationships, attributes, and types.

3.8 DESIGN ALTERNATIVES

An alternative to any sort of central registration of type information is to have every tool or tool suite carry its own type information, similar to the PCTE approach. It is not clear, however, how this aids overall database integrity, how this promotes I and T, or what problems arise when interchanging data among tool suites.

3.9 RECOMMENDED DESIGN RESOLUTION

TBD

3.10 PERFORMANCE IMPACT OF RECOMMENDED DESIGN

TBD

3.11 UPWARD COMPATIBILITY FROM CAIS 1 TO CAIS 2 OF RECOMMENDED DESIGN

TBD

3.12 SECURITY ISSUES FOR RECOMMENDED DESIGN

TBD

3.13 IMPLEMENTABILITY OF RECOMMENDED DESIGN

TBD

3.14 OTHER ISSUES WITH RECOMMENDED DESIGN

TBD

ISSUE SUMMARY

ISSUE REPORT REVISION HISTORY

ISSUE 4.0 -- I/O Priorities

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	6/30		First Draft

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

MAJOR ISSUE: What data are to be controlled by the EMS typing mechanism? How is that to be accomplished?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
4.1/0	Data Representation Issues	
4.1.1/0	Levels of Data Representation	
4.1.1/0	RESOLVED	Data representation is a hierarchy of types with bit string at the base, then byte string, character string, then day, month, year, then date, etc.
4.1.1/0	RESOLVED	Entity contents and attribute values are the only objects subject to explicit representation typing. Everything else is controlled by Ada typing via the part of the CAIS specified in Ada. Contents are this way too; but interoperability may require some sort of control over contents representation.
4.1.1/0	RESOLVED	To promote intertool interoperability, the CAIS will define 3 "standard" content representations: 1. Sophisticated text, with ASCII text being a sub-part, 2. Graphic representation, with sophisticated text as a sub-part, and 3. Descriptor for files created by Ada. Direct and/or Sequential IO Users are free to create their own file formats in addition to these.
4.1.1/0	OPEN	Do we support aggregate representation types?
4.3/0	Existing Standards	
4.3/0	OPEN	Which, if any, existing standards can be used for CAIS standard representations?

SOPHISTICATED TEXT	GRAPHICS
Microsoft Windows	GKS
DGIS	CORE
PostScript	VDI/VDM
(Interpress)	NAPALPS
Apple QuickDraw	PHIGS

ISSUE SUMMARY

MAJOR ISSUE: Can I/O and interprocess communication be unified?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
4.4	Commonality of I/O and Interprocess Communication	
4.4.4/0	Compatibility with CAIS 1	
4.4.4/0	OPEN	Can the notion of queue nodes be subsumed?
4.5/1	Device Drivers	
4.5/1	RESOLVED	CAIS will have user-installable code fragments called device drivers (DDs). DDs will be used to convert device independent data representations to device dependent data streams with explicit control information. DDs are device-dependent, but host-independent; they can be (should be) written in Ada.
4.5/1	OPEN	What is the design of the tool-DD interface?
4.5.1.3	A Portion of CAIS Outside the Node Model	
4.5.1.4/0	A Special Kind of Device Node	
4.5.1.4/0	OPEN	Do DDs and CDHs appear in the DB? If so, what form do they take?
4.5.1.5/1	Composable DDs	
4.5.1.5/0	OPEN	It should be possible to compose DDs so that one can act as a terminal window manager dispatching data received from other DDs, e.g., one for each window.
4.5.2/0	Recommended Design	
4.5.2/0	RESOLVED	In order to promote portability of DDs, Host dependent code will be placed in CAIS Device Handlers (CDHs). CDHs are device and host dependent.
4.5.2/0	OPEN	What is the design of the DD-CDH interface?
4.5.2/0	RESOLVED	To accomodate asynchronous event handling, both the DD-Tool interface and the DD-CHD interface will supply a software interrupt signalling mechanism. The tool writer may elect to use this mechanism, or Ada tasking, as deemed appropriate to the job at hand.
4.5.2/0	OPEN	How are DDs and CHDs installed? Is this issue outside the CAIS?

4.5.7/0	DDs vs Gatekeepers		
4.5.7/0		OPEN	Are there special DDs (called gatekeepers) used as agents for connection to remote services in a distributed CAIS?

CHAPTER 4

I/O PRIORITIES

This chapter discusses data representation issues, the user view versus the tool view of the I/O interface, interprocess communication, and device drivers (DDs).

4.1 DATA REPRESENTATION ISSUES

This section addresses issues associated with data representations and typing of data representations within CAIS 2. The issues identified help define the basic data representation capabilities of CAIS 2. Since the EMS typing implementation has not been established and is considered to affect the selection of the data representation implementation, we are deferring this topic. The remainder of this section highlights the categories of issues and current implementation thinking that will be addressed with regard to data representation once we have a preliminary implementation approach for EMS typing.

4.1.1 Level(s) of Data Representation Included in CAIS 2

Specific topics to be covered include:

a. Discussion of the extremes in representation which go from bits to a software project documentation tree.

b. Data representations for attributes.

c. Data representations for entity contents.

d. Are data representations associated only with elementary values of attributes/contents or can they be associated with a "structured" collection of nodes, relationships, and attributes?

4.1.1.1 Design Alternatives

1. Have data representation for attribute and contents instances only (files, DD, etc).
2. Have a complex data representation scheme that allows us to treat a complex connection of nodes and relationships as a single aggregate.
3. Use 1 above, but allow the Entity structure to form complex items within the constraints of the EMS typing scheme.
4. Predefine all data representations, except contents.

4.1.1.2 Recommended Design Resolution

Use approach 1. Approach 3 will work if CAIS builds on 1.

4.1.1.3 Performance Impact of Recommended Design

TBD

4.1.1.4 Compatibility with CAIS 1

Extends the CAIS 1 unstructured contents to a structured one.

4.1.1.5 Security Issues for Recommended Design

None.

4.1.1.6 Implementability of the Recommended Design

a. Data representations could be implemented with static "Ada type managers" which a tool imports. This would only require us to have a collection of nodes that provide the predefined representations. See Appendix A.

b. Implementing a more dynamic linking capability could be done on bare hosts but would be difficult on some piggy-backs. The performance impact of a dynamic linking capability could be considerable.

4.1.2 Specifications of Data Representations in CAIS

Specific topics to be covered include:

a. How much implementation detail is needed to make a data representation transportable?

b. Can we create usable abstract data types to support data representations?

c. Does Ada have the expressive power to allow for transporting of data representations?

d. Can a general data translation tool be constructed?

4.1.2.1 Design Alternatives

1. Only provide a set of predefined types that would be part of the standard.
2. Provide a mechanism for user defined representations in the standard and define predefined representations using it. (This is what Ada does for I/O.)
3. If we implement data representations on contents of files only and use static "Ada Type Managers" or abstract data types to represent them then we can implement 2 without having to specify a user defined representation mechanism in the standard.

4.1.3 Recommended Design Resolution

Design alternative 3 above.

4.1.4 Performance Impact of Recommended Design

TBD

4.1.5 Compatibility of Recommenaed Design with CAIS 1

A simple extension of the way data is represented in CAIS 1.

4.1.6 Security Issues for Recommended Design

TBD

4.1.7 Implementability of the Recommended Design

TBD

4.2 USER VIEW VS. TOOL VIEW

Should we directly support the APSE user view or only a tool view of the interface with predefined representations?

Specific topics to be covered include:

a. Do windowing and user interfacing issues become part of a data representation?

b. How do we factor in interactive devices like a mouse, etc? A "sophisticated text" representation should include some capability to support a wide range of commonly used interactive input devices. Having a "sophisticated text" type handler would satisfy this requirement.

4.2.1 Design Alternatives

1. Limit predefined representations to a very elementary level (bit streams, bytes, character streams, list type).

2. Expand 1 above to include a predefined repr^esentation of text that would appear on a multiple window worksᵗation.

3. Expand 2 above to include some level of mixed text/graphics support.

4.2.2 Recommended Design Resolution

Design alternative 3 - "sophisticated text."

4.2.2.1 Performance Impact of Recommended Design

TBD

4.2.2.2 Compatibility of Recommended Design with CAIS 1

TBD

4.2.2.3 Security Issues for Recommended Design

TBD

4.2.2.4 Implementability of the Recommended Design

TBD

4.2.3 Other Design Issues

How do we achieve transportability of data representations? If we use static Ada type managers and Ada representatic1 standards to implement the data representations of entity contents, then we can port a data representation by moving the tools that use a representation and the package that defines the representation.

All representations should fit into a representation hierarchy. At the highest level we should be able to manipulate all contents as bit streams. From bit streams we can build other data representations such as byte streams, or more complex representations. Using byte streams we should be able to build character streams, etc. This data representation hierarchy will allow us to transport a representation at several levels depending on the needs of the user and the conversion facilities.

Can the concepts of data representation be expressed as a limited number of conceptual models?

Can a representation be distributed?

4.3 EXISTING STANDARDS

What existing standards should we consider as a basis for predefined data representations?

Discussion of available and de facto standards for data representations.

4.3.1 Design Alternatives

1. Use an industry graphics standard for sophisticated text. Ones to consider GKS, CORE, VDI/VDM or NAPALPS.
2. Use a de facto standard like Microsoft or SUN "Windows", DGIS, Postscript, or Apple Quickdraw for sophisticated text.
3. Use a hybrid from several standards.

4.3.2 Recommended Design Resolution

Open.

4.3.2.1 Performance Impact of Recommended Design

TBD

4.3.2.2 Compatibility of Recommended)esign with CAIS 1

TBD

4.3.2.3 Security Issues for Recommended Design

TBD

4.3.2.4 Implementability of the Recommended Design

TBD

4.4 COMMONALITY OF I/O AND INTERPROCESS COMMUNICATION (IPC)

Currently, CAIS handles process control, I/O, and interprocess communication as largely separate functions. While examining the primitives required to support DDs, a strong similarity was noticed between the type of communication required between an DD and a tool, and CAIS interprocess communication. The possibility that I/O and interprocess communication could be represented by the same basic model was pursued. This approach could reduce the number and complexity of CAIS interfaces as well as concentrate many asynchronous control issues in one place. This issue report discusses our current thoughts on such a general model for I/O and communication.

An overview of such a model is described here. Its primary goals are:

a. simplification of the number of system interfaces, and

b. transparency to the application program of file I/O and interprocess communication.

Interprocess communication (IPC) is the sending/receiving of messages between two or more processes and the handling of the associated control

and status information for the communication channel(s). Processes communicate via "ports" to each process. A process may have several ports active simultaneously. A port may be specified as read or write and synchronous or asynchronous. An asynchronous port would have an independent channel for status, control, or error reporting.

Processes should be able to create and delete ports both statically or dynamically, subject to type restrictions. The connection between ports would be modeled with EMS relationships, where the relationship type could determine the kind of communication protocol between ports. The relationships could form a network extended to include intermediate processes such as DDs and Gatekeepers between the sender and receiver, tranparent to both of them. Unifying device-oriented I/O and process-oriented I/O allows the substitution of a process for a device with no change in the code, or even relinking. This is a very powerful feature useful for debugging as well as for dynamic composition of processes into larger tools. UNIX pipes and command processor pipelines are a rough paradigm for this capability.

Asynchrouous events would be handled in Ada by an Interrupt Handler which would receive software-generated interrupts. A skeleton of such a handler is being prepared. The handler would be compiled with those processes requiring asynchronous control.

File I/O would resemble IPC. Data would be transferred to and from files by sending requests to a file handling process. No-wait I/O would correspond to asynchronous messages.

Exceptions would be handled by a common mechanism, whereby software interrupts would be sent to a process, and the process interrupt handler would raise the appropriate exception. Common exception handlers could be provided as library routines.

This model has implications for the structure of every CAIS program. Each program would have to deal with ports, exceptions, and status/error messages. The addition of a CAIS method for dealing with non-exception interrupts is a valuable addition to the repertoire of tool writers. It may prompt the reconsideration of some CAIS events now classified as exceptions to the lesser classification of status/error response.

Integral to the unified I/O and IPC model is a form of preferably dynamic type checking of the information being communicated. Omitting checking would be similar to the UNIX method, where all processes expect byte streams. CAIS should make available adequate information in each port definition to ensure sending and receiving ports are likely to understand the format of communicated information. This is discussed further later.

A strawman proposal is being developed for this unified model.

4.4.1 Design Alternatives

The alternatives are to keep I/O and interprocess communication as separate concepts, interfaces, and implementations, or to unify these in CAIS 2.

4.4.2 Recommended Design Resolution

A Logical Device Driver should provide for both synchronous data flow and asynchronous event processing. Most modern, highly interactive tools will require asynchronous signalling for effectively controlling terminals equiped with mice, light pens, etc. Even common keypad editors require character-level signalling, as do the break-in (attention) functions of most conventional operating systems. Though not explicit, this need is acknowledged in RAC 6.3F, 6.3G, 6.4B, 6.4C, 6.4E, 6.4L, and 6.4M of the September 1985 edition. Although it might be possible to hide all asynchronous information from a tool, it is viewed as unneccessarily limiting to disallow direct no-wait I/O and prompt response to device interrupts. Ideally, an DD would have either (or both) synchronous or asynchronous data, control as well as status paths with a client tool.

This type of interface is exactly the same as is required to support interprocess communication, specifically, a blocked and unblocked SEND_MESSAGE corresponding to wait and no-wait WRITE, and a blocked and unblocked READ_MESSAGE corresponding to wait and no-wait READ. Control and/or status paths may be used to adjust queue sizes, inquire about message status, confirm message postings, etc.

The use of a unified interprocess communication and I/O model is recommended.

4.4.3 Performance Impact of Recommended Design

No significant performance impact has been identified at the design level.

4.4.4 Compatibility of Recommended Design with CAIS 1

The functionality of queuing nodes would need to be re-examined. It would be preferable for solo queues to be transparent to the tool makers. They could be implemented by CAIS as one attribute of the interprocess relationship. Mimic and copy queues could remain as CAIS utility processes.

Changes to the model could be hidden in CAIS, such that CAIS 2 resembled CAIS 1 more closely, or they could be made directly visible to the users. CAIS 1 and LRM Chapter 14 I/O could be supported as a utility packa̵gc cf CAIS 2 or perhaps

as overlcaded interfaces. CAIS 2 would have a more primitive, but more powerful model which would yield greater flexibility and protection.

4.4.5 Security Issues for Recommended Design

This proposal may introduce some new security problems not present in the CAIS 1 system, especially if communication is allowed between processes started by different users.

4.4.6 Implementability of the Recommended Design

Conceptually similar designs have been implemented elsewhere, such as IBM's Network Implementation Language.

4.4.7 Other Design Issues

UNIX pipes and file re-direction can be supported by the proposed model. UNIX-like text communication among tools is one possible subset of the allowable data formats. This should simplify conversion of UNIX-based tools to the CAIS. It might be possible to host UNIX, or at least major UNIX environment services, as a subset of CAIS.

CAIS should allow multiple communication paths to be open simultaneously. This should be an obvious requirement to support multiple I/O devices from a single process, but applies as well to independent ports for standard input, standard output, and standard error. A tool may choose to handle data, control messages, and errors on different ports rather than to sort and priortize these from a single input. It is not yet determined whether multiple writers to a single port should be allowed directly, or whether such a capability must be supported by an explicit queue process. We are leaning toward a CAIS-wide rule that only one writer be allowed per port, but that multiple writers be supported by queue processes transparenc to the tools. (They do not need to have knowledge of the queue name, or inquire of the receiver the queue's identity. The CAIS services would handle this when desired.)

Some of the proposed features of CAIS 2 are known to be supported by other systems. A representative set of these should be examined to learn from their experiences. These include UNIX 4.2 BSD with Sun's Network File System, UNIX V, Xerox Network System, and IBM's Network Implementation Language. This list will be expanded as appropriate.

4.5 DEVICE DRIVERS

A logical device driver is the interface between a tool and the underlying operating system and/or physical I/O devices. An DD may represent a very complex device, such as a GKS workstation, or a simple interface, such as a card reader. CAIS DD interfaces are largely standardized to allow DDs to be written in Ada and moved between CAIS implementations. DDs are bound to processes during execution. DDs, however, are device-dependent. DDs are the repository of device-dependent code, enabling tools to be as independent as possible of devices. It is expected, however, that some tools will rely on conventions which establish interfaces between DDs and tools. We expect at least two such conventions to be part of the CAIS 2 Standard: the Sophisticated Text Representation and the Graphics Representation. It is intended that other conventions will become de facto standards as the technology evolves. This laissez-faire approach will allow CAIS implementations to adapt relatively quickly to new (or old) devices as needs materialize. Well-accepted standards can become de facto standards as the CAIS matures.

An DD is a functional concept which may map to several physical implementations. An DD could be a CAIS process, a group of CAIS processes, a portion of CAIS outside user control and/or the node model, a special kind of device node, or a group of procedural interfaces bound with a process. The CAIS 2 design will elaborate one or more of these alternatives.

The impact of DD design on other aspects of CAIS could be substantial. I/O forms a large portion of CAIS 1. DD interfaces may impact other CAIS services and will certainly influence the style of tool writing.

4.5.1 Design Alternatives

The alternatives are to implement an DD as a CAIS process, or as a group of CAIS processes, or as a special kind of device node, or not to specify the implementation.

4.5.1.1 CAIS Process

If each DD were a CAIS process, then DDs would be portable. Operations on DDs would be consistent with operations on any node. However, there is a device-dependent and/or machine dependent function to an DD which may disallow such a simple placement in the CAIS model. One proposal is to separate the machine/device dependent code from the rest of the DD. This would be put into the equivalent of OS device handlers named CAIS device handlers (CDH) which performed only low-level functions such as READ, WRITE, SEND CONTROL, and READ STATUS. The CAIS-side interface to CDHs would be well defined, but the machine/OS side would be entirely implementation dependent. DDs would be portable as long as the importing CAIS implementation had an appropriate CDH.

4.5.1.2 Group of CAIS Processes

Since DDs may represent very simple or very complex devices, it may be useful to build sophisticated DDs from simpler ones. This would mean that a group of processes may form an DD. Some of these processes may be DDs themselves which would be available to tools desiring more primitive functions or more direct control over devices.

4.5.1.3 A Portion of CAIS Outside the Node Model

If DDs were outside of the standard, they could be implemented with complete freedom as long as they met the interface specifications. However, this approach does not promote the portability of DDs.

4.5.1.4 A Special Kind of Device Node

If an DD were a special kind of device node, it would not necessarily be portable, but its interface would be consistent with other I/O interfaces in the system.

4.5.1.5 Composable DDs

DDs imply a certain partitioning of device types into related devices with common characteristics, e.g., unit record, random access, dumb terminal, etc. Can DDs be layered on one another to implement more sophisticated classes from the simpler ones? If layering of DDs is to be allowea, CAIS will have to define either static or dynamic methods of combining DDs.

One idea for a sophisticated text (text and graphics) DD is to use a multi-layered DD. Graphics raster, graphics vector, form text, plain text would all be a part of the DD. It would (1) use whichever level was appropriate based on the physical device available and (2) utilize the primitive functions to implement some of the simple features of the higher-level devices.

The alternative to such layering is to have each DD be unique. The primary impact of this is in the binding of DD pieces. The former encourages a dynamic binding, the later a static building of an DD from a library (which may or may not reuse lower level DD routines).

4.5.2 Recommended Design Resolution

The division of DDs into two parts (one portable and one system-dependent) is recommended. Implementors would have the freedom to develop CDH interfaces that were either procedural or process-communication oriented. These could exist as CAIS entities, either libraries of procedures or process nodes. Both DD interfaces would be CAIS defined, enhancing the portability of CHDs. It is not clear that the division between host-dependent and host-independent parts can be cleanly achieved.

If DDs are either CAIS processes or portions of CAIS processes, the asynchronous nature of some device I/O cannot simply be buried within CAIS. A CAIS-wide mechanism for handling asynchronous interrupts must be developed. The Ada exception mechanism alone does not provide the information transfer needed to support this type of communication. Standard methods of handling error and normal status information are also needed. Such methods should allow tools to be programmed so that they may return to the point of execution after interrupt. (A separate issue report is planned on Interrupts and Exception Handling.)

The ability to bind DDs dynamically from more primitive DDs is considered to offer the most flexibility to the implementors, and to encourage the reuse of more DD code. It does not prohibit an DD from being a single, indivisible unit. The dynamic linking is most readily accomplished when DDs are implemented as processes. The static binding could be utilized in a procedural library which accessed the CDH directly.

4.5.3 Performance Impact of Recommended Design

The division of an DD into two portions may impact its performance, but should help portability. It is not possible to accurately quantify any such performance penalties at this time. It might be allowable to have an optional higher performance version of an DD which bound the two portions together, as long as the standard implementations were also available.

Procedural interfaces would almost certainly be implemented as conversions to the process-communication interfaces. Again this is trading off some performance for the goals of portability and interoperability. (Remote DDs would be available across processor boundaries if process-communication interfaces were followed.)

The layered approach is predicted to be less efficient, but more versatile. The direct option is not prohibited and would be recommended in places where required for additional performance. (Intuitively it would seem that the additional layers would slow down an DD; however, it might be possible for unique processors to be assigned to each portion of a multiprocess DD, resulting in faster operation.)

4.5.4 Compatibility of Recommended Design with CAIS 1

CAIS 1 I/O services could be implemented as utility package using the CAIS 2 paradigm. There could be special DDs supporting CAIS 1 I/O.

4.5.5 Security Issues for Recommended Design

The design poses no new security issues; in fact, by having DDs within CAIS it utilizes CAIS security. If they were "special" beyond CAIS items, security for these items would have to addressed in their design and implementation on each system. Allowing on-the-fly installation of DDs may not be allowed in secure CAIS implementations.

4.5.6 Implementability of the Recommended Design

It is suggested that the recommended design be implemented using the basic CAIS 2 interprocess-I/O communication model discussed below. The skeleton of an Ada implementation for handling asynchronous I/O is in Fig 4.3.7. (TBD)

4.5.7 DDs vs. Gatekeepers

Just as DDs are a window into the machine/device dependent world from CAIS, gatekeepers are a window into the distributed world. Although a great deal of additional processing is invoked to communicate reliably across processor boundaries, CAIS may either strive to make this transparent to tools or require them to explicitly invoke special communication mechanisms.

Is there a distinction between a gatekeepei and an DD? Should there be? If a gatekeeper is considered to be a variety of DD then the interfaces with tools and the relationship of gatekeepers to CAIS is consistent with those developed for DDs. If there is some uniquely distinguishing property of gatekeepers such that they cannot be reasonably treated as DDs, then their tool and CAIS interfaces must be separately determined.

CAIS 2 should hide the extra processing involved to utilize remote devices and/or communicate with remote processes from the individual tools, except those "tools" which are specifically involved with the management of such remote communications.

An implementation is envisioned where the primary distinguishing characteristic of remote access at the tool level is in the naming conventions. This topic will be covered in more depth in the distribution issue report.

ISSUE SUMMARY

ISSUE REPORT REVISION HISTORY

ISSUE 5.0 -- Security and Access Control

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	6/30		First Draft

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

ISSUE SUMMARY

MAJOR ISSUE: Is a secure CAIS implementation possible?

SECTION/REV REFFRENCE	STATUS	DESCRIPTION OF ISSUE
5.0 Security and Access Control		
5.0/0	OPEN	Does CAIS need a security policy or a security policy model of its own? Can such a policy be complete, or partial? How can such a model be reconciled with the security model of an underlying TCB?
5.0/0	OPEN	Can the CAIS design be mapped to a TCB supporting the Bell-LaPadula security model?
5.0/0	OPEN	Does the CAIS design contain any unacceptable security leaks?
5.0/0	OPEN	Can the present CAIS security mechanism be simplified?

CHAPTER 5

SECURITY AND ACCESS CONTROL

This section describes background security issues, security implications of distribution, and access control on all nodes versus on all leaves.

5.1 BACKGROUND SECURITY ISSUES

This section provides insights into some computer security issues as they pertain to the development of a general purpose operating system that may be used as a TCB for CAIS Version 2. Topic areas include: (1) an investigation into what features and services are required within the TCB kernel, (2) how that determination is made, (3) the inclusion and location of trusted functions versus performance and complexity tradeoffs, and (4) the commercial products currently available that could provide CAIS with a B3/A1 level of trust.

Although this material is not strictly relevant to the interface design itself, it is supplied in support of later discussion.

5.1.1 Security Requirements for a B3 System

The level of trust necessary to thwart the risks associated with the implementation of the CAIS has been established at the B3/A1 level, as defined in the Department of Defense Trusted Computer System Evaluation Criteria (CSC-STD-001-83); hereafter referred to as "the criteria." A focal concept in developing an upper-end criteria system is the identification of the Trusted Computer Base (TCB). The B3 criteria require that the TCB be composed of a small and simple subset of the system that is responsible for ensuring security of the total system.

The TCB includes both the security kernel, which implements the reference monitor concept and manages the physical resources, and the trusted subjects, which support refinement to the fundamental policy of the system. It is the reference monitor that provides an underlying security theory for conceptualizing the idea of protection in B3 systems. All active entities in a reference monitor, such as users or processes, make reference to passive entities such as pages or segments of memory using the prescribed set of access authorizations. The implementation of the reference monitor or the security kernel consists of both hardware and software elements. A primary requirement of a B3 system is that the implementation of a security kernel consist of a small and non-complex segment of an operating system. Further, the security kernel is developed at a low level base where basic security functions are provided in a highly reliable way. Extensive operating system functions will be built on top of this skeletal base. User support is also built upon the security kernel and operating system.

5.1.2 Security Policy and Policy Model

In order to satisfy the B3 requirements of the criteria, a mathematical security model that represents the security policy enforced by the system must be developed and proven consistent with its axioms. In addition, the TCB, which is the totality of protection mechanisms within CAIS, must be founded on a formal security policy model that applies to all subjects and objects in the system. Before a CAIS security policy model can be defined, a precise statement of the security policy must be identified. The statement stipulates the security policy requirements, and it must be mutually agreed upon by the developer and the government. The security policy statement should be sufficiently broad so as not to constrain the design and implementation of CAIS. However, it should capture the essence of what it means for the CAIS to be secure. An important question is whether such a policy or part of such a policy must be given in the CAIS standard or whether it should be defined by implementors.

5.1.2.1 Security Policy Statement

The security policy statement is a general description of the technical requirements that define the security policy for the system. The first step in its development is to identify the requirements that govern the security functionality of the system. As the CAIS security-relevant requirements are analyzed and the CAIS security architecture is determined, the system security policy can be allocated to security functions that are performed at the subsystem level. The resulting subsystems would each have security functionality that, when taken together, provide the desired system-wide security behavior.

The CAIS security policy statement should address the following:

a. Access Authorization. The portion of the security policy that describes how access control decisions are made. (This probably must be specified in CAIS as it pertains to CAIS objects.)

b. Accountability. That portion of the security policy that addresses user identification and authentication and the requirements for a security audit trail. (The authentication issue is probably outside the realm of the standard.)

c. Assurance. That portion of the security policy that addresses the development methodology, testing requirements, and the protection of the system security features. (Largely outside the standard.)

The security policy statement is the basis for the definition of the mathematical security policy model. The security policy statement must be written in such a way that the components of the security model can correspond to it. For example, if the term user were used in the policy statement it should correspond to the term subject in the model.

5.1.2.2 Security Policy Model

Formal mathematical methods exist for studying and analyzing the information accesses that are authorized by the system under certain assumptions given by the access rules (access authorization). Formal techniques are not available for representing the other portions of the security policy (accountability and assurance).

Therefore, the term "mathematical policy model" should be considered to denote a mathematical model of the access authorization portion of the CAIS system. Based on the B3 criteria, CAIS will need to provide two types of access policy: (1) non-discretionary, and (2) discretionary. A non-discretionary policy contains mandatory security rules that are imposed on all users. A discretionary policy contains rules that can be specified at the option of each user.

The rules of the CAIS security model must address both discretionary and mandatory policies. It is suggested that the CAIS security policy model be derived from the Bell & LaPadula model. The Bell & LaPadula model is currently the only security policy model accepted by the National Computer Security Center (NCSC). This model provides rules for preventing unauthorized observation and modification of information. These rules are described in the sections that follow.

By representing the TCB as a finite state machine, the rules of the policy model define allowable transitions from one secure state to the next. Within the model, each active entity or subject and each passive entity or object is given a security identifier termed an access class. The access class of each subject and object is compared at each state transition to determine whether the subject has access rights to an object.

By organizing the access classes in the form of a mathematical structure called a lattice, a wide range of potential policies can be supported. The lattice defines the relationships among access classes, allowing the determination of whether one access class is less than, greater than, equal to, or not comparable to another. An example of a lattice as it pertains to the hierarchical government security classification is shown in Figure 5-1.

OBJECTS

		U	C	S	TS
SUBJECTS	U	Read/ Write	Write	Write	Write
	C	Read	Read/ Write	Write	Write
	S	Read	Read	Read/ Write	Write
	TS	Read	Read	Read	Read/ Write

Figure 5-1. Government Security Classification lattice

5.1.2.2.1 Mandatory Security

Two security properties are fundamental to, and are described, in the Bell & LaPadula model. They are the simple security condition and the *-property ("star property"). The simple security condition states that a subject can not observe (read) the contents of an object unless the access class of the subject is greater than or equal to the access class of the object. The simple security condition prohibits subjects from directly viewing information for which they are not cleared. The notion of the simple security proper is illustrated in Figure 5-1, where the "read" designator represents a subject-object relation sufficient for a subject to observe an object. The *-property, on the other hand, prevents all illicit, indirect viewing of objects. It states that a subject cannot modify an object unless the subject's access class is less than or equal to the access class of the object. The *-property is also illustrated in Figure 2-1, by the "write" designator. The purpose of the *-property is to deal with the problem of "Trojan Horse" software. A Trojan Horse, as the name indicates, is software that appears to do some useful function, but while performing that useful function it performs an illicit act. Any generally used software utility has the potential for handling a user's file in a manner the user did not intend. Examples of targeted utilities are text editors or compilers.

Enforcing the *-property within access control mechanisms ensures security cannot be compromised through a Trojan Horse.

5.1.2.2.2 Discretionary Security

Unlike mandatory access control, the discretionary access control rules of the Bell & LaPadula model provide a protection policy that distinguishes different users within the same access class. An access class is composed of both hierarchical (Unclassified, Confidential, and Secret) and non-hierarchical (compartmented) clearances. The discretionary rules allow authorized users and programs to arbitrarily grant and revoke access to information based on users names or other information. The military "need-to-know" controls are an example of discretionary access control. Although a B3 system provides for both mandatory and discretionary access controls, where discretionary controls provide a finer access control granularity, in no case may discretionary controls override mandatory controls.

5.1.3 Security Kernel

The number and type of mechanisms that must be built into a security kernel are a direct result of the mandated security policy. A mathematical model is a powerful design tool for translating the requirements of an abstractly described security policy into a precise representation of the behavior of the corresponding security kernel. The mathematical security model describes the initial secure state of the system with respect to the policy being considered and carefully defines a set of functions or rules for changing the state of the system. This means that if the model can be proven to indeed represent the behavior of the security policy, then no use of the kernel can cause a violation of the security policy. In addition, by building the kernel to comply with the basic requirement that it always be invoked for an information transfer to take place, non-kernel software is consequently prevented from violating security policy. Therefore, the model is used to dictate what must and must not be included in the kernel.

5.1.3.1 Kernel Characteristics

The security kernel is the hardware and software implementation of the security policy model. To be effective, the security kernel must meet three engineering principles:

a. Complete mediation of all access (by subjects) to information (objects).

b. Isolation and protection of the security kernel itself from penetration and subversion.

c. Verifiability of the consistent and reliable enforcement of the access authorization prescribed in the security model.

The security kernel interface consists of a set of subroutines that can be invoked by other programs. It can be invoked not only by applications, but also by the rest of the operating system (supervisor). The kernel and only the kernel controls and manages all hardware components that process and store information.

The first principle, (a) above, can have significant implications for CAISes implemented over a TCB. This would imply that either the CAIS have its own direct access to the reference monitor, or have its own reference monitor, or else the traffic across the host operating system boundary will be very high. In the latter case, access to each node, attribute, and relationship will have to be routed individually via some reference monitor.

5.1.3.2 Kernel and Supervisor Functions

To successfully implement a kernel-based operating system, architectural and engineering considerations must be taken into account that are not typically found in non-kernel based operating systems. As mentioned earlier, the security kernel must be provided by a relatively small and simple subset of the operating system functions. The kernel primitives are the interface of this subset to the rest of the operating system, generally referred to as the supervisor. The supervisor primitives provide the operating system functions used by the applications (see Figure 5-2). An operating system is broken down into functional areas, such as process management, file system management for segments, and I/O control. Within each area, some functions are security relevant or policy relevant and must be implemented within the security kernel; others are not. The rules or security properties associated with the security model help to clearly dictate which of these functions are security relevant. The security kernel must handle the parts of the operating system that manage resources. Examples are: (1) memory, (2) disk space, and (3) multiple users. These parts are implemented witbin the security kernel because the security model requires that these resources be virtual to hide their locations from untrusted, non-kernel software. Functions that provide useful common utilities, but do not manage anything shared among users, are outside the scope of the security policy and are placed in the supervisor.

The security model could be used solely to determine what functions belong in the kernel. However. real-life problems such as performance and overall system complexity require moving functions into the kernel which are not security relevant, which would result in its increased size and complexity.

```
            Applications
        ------------------  Level 3
Op.         Supervisor
System  ------------------  Level 2
            Kernel
        ------------------  Level 1
            Hardware
```

Figure 5-2. 3 State Machine

5.1.3.2.1 Kernel Size/Complexity vs. System Complexity/Performance

Although it is important that the developer minimize the size and complexity of the security kernel, tradeoffs need to be considered. That is because functions included in the operating system base to enhance performance or increase the convenience of writing software above that base are not needed for security purposes and consequently are not included in the kernel design. Usually, security kernel designers exclude this software despite potentially increased performance overhead or greater non-kernel software complexity. Therefore, there is a need to consider moving functions that are not security model driven into the kernel. For example, it may be difficult separating the operating system's file-name interpretation mechanism, which may not be security relevant, from the kernel's file management system.

5.1.3.2.2 Kernel Functions

For the purposes of this paper a list of kernel functions can intuitively be defined, assuming a Bell & LaPadula type model. Other functions may need to be considered once a more refined security model is developed. Concurrently, performance and complexity issues must be addressed. At a minimum, the security kernel must implement all functions controlling resource management, process scheduling, memory management, interrupt handling, and auditing. The security kernel must also function as the software portion of the reference monitor implementation in that it controls access to objects in accordance with its embedded security policy. The security kernel must support memory segmentation, devices, and processes. Each of these objects (passive entities) must be uniquely distinguished from one another, and their identities must be immutable throughout the life of the system. The kernel must also maintain two types of access information in the system. The access information consists of both hierarchical and non-hierarchical attributes (mandatory controls), and discretionary information that includes read, write, and execute permissions controlled by the owner of the information.

5.1.3.3 Kernel Features

Specific mechanisms have been proven necessary to support a kernel-based operating system in four general architectural areas. They are:

a. Explicit Processes

b. Explicit Segments

c. Execution Domains

d. I/O Mediation

5.1.3.3.1 Explicit Processes

A process can be thought of in terms of the activity of a processor in carrying out the computation or I/O process specified by a program. Ultimately, a process serves as a surrogate for a user. For information protection to be meaningful the environment must support multiple processes. The user's identification and access class must, therefore, be represented within the system as nonforgeable identifiers tied to each processor. These identifiers are the basis for making access control decisions within the system. The requirement for the kernel to support multiple processors means that the kernel must have the capability to save and restore the representation of a process in execution (state of process). In addition, the architecture must provide for the saving and restoring a definition of the accessible information (i.e., the address space) distinct for each process. The address space will typically be represented by a set of descriptors.

5.1.3.3.2 Explicit Segments

The reference monitor abstraction of an object is realized by memory, and this realization is constrained by the principle of complete mediation. To completely mediate all access to memory, it is clear that some fundamentally interpretive mechanism is needed. For I/O processes, this mediation can be provided by the kernel software if only the kernel software can issue I/O requests (e.g., if I/O uses privileged instructions only available to the kernel). For other processes, purely interpretive execution is not practical and virtual memory is a common mechanism for accomplishing the needed mediation. Some form of descriptor is used to control the access to memory and there must be mo means for a process to access memory without a descriptor. With a reference monitor, all information within the system memory must be represented in distinct, identifiable objects referred to as segments. The hardware supported segmentation of virtual memory is the basis to support this concept. Each segment is identified by a descriptor that controls the virtual address mapping hardware. The descriptor contains some logical attributes as well as physical-based address and segment size to distinguish each segment. The descriptors, of course, must be managed by the security kernel, although much of the actual mediation of the reference monitor is performed by the address mapping hardware. The security kernel software enforces the reference monitor authorizations by controlling the access mode specified in the descriptors for the segments of each process.

5.1.3.3.3 Execution Domains

Execution domains are essential to the isolation and protection of the security kernel mechanism. The total address space of a process includes the program and data of the security kernel since these must be accessible when the kernel is invoked. Clearly, these must also be accessible when the security kernel subroutines are invoked. It is also clear that the kernel requires a distinct execution domain such that a process can access some object (segment descriptor) only when invoked from the kernel itself. To provide kernel protection, at least two separate states or domains are needed. The kernel would reside in a privileged domain and the supervisor and the applications software would reside in the other. Although the dual domain approach would be adequate from a security perspective, it would deviate from the more traditional approach of designing an operating system where the entire operating system resides in a single privileged domain and the applications programs reside in a less privileged domain. Therefore, in order to provide for the requirement of having the security kernel placed at the most privileged domain and provide for a supervisor/applications separation, there is a need for three hierarchical domains.

5.1.3.3.4 I/O Mediation

I/O operations can generally be accomplished in two different ways. The first way is to have software explicitly execute an I/O instruction to transfer each unit of information between I/O device and memory. In this case, the reference monitor would view the I/O devices as objects.

The kernel would control access to these devices. In other words, I/O instructions would be required to reside in the most privileged domain (kernel), and user and supervisor software would invoke kernel functions to perform I/O services.

A more complex approach to I/O operations relies on independent I/O processors. Once activated by the CPU, each processor asynchronously transfers information between devices and memory. This transfer of information is accomplished by the I/O program residing in memory or in the I/O processor itself, where the programs are given parameters such as buffer and device addresses. In this case the kernel must consider the I/O programs in execution as subjects and it must control access to memory by these subjects in the same manner as it controls access to memory by any other subject (i.e., user or process). As in the first I/O method, the conventional approach to handling this access control is for hardware to limit initiation of I/O processors to the most privileged domain. Therefore, a user must request I/O in the form of a kernel function call, where a check of the I/O program or parameters is performed to ensure that both I/O devices and memory segments contained in the I/O buffers are accessible to the user. Because I/O processors usually lack multiple domains, I/O implemented must be in the kernel domain. Therefore, the processor uses physical memory addresses supplied by the kernel and the kernel translates virtual addresses to physical addresses. Because of the complexity of handling I/O, a hardware architecture that allows direct user or supervisor access to I/O is desirable. This architecture would provide some form of descriptor to control access to the devices, in a manner similar to the use of memory descriptors. In addition, for the I/O processor to effectively operate outside the kernel by accessing virtual memory on behalf of the user, there is a need for descriptor controlled access to memory by the I/O processor.

5.1.4 Trusted Software

Most systems require a security policy that is more specific to their needs than the one achieved by a security kernel. A more tailored policy can be exercised on a limited basis for infrequent but essential operations. In order for the operating system to support such an extended policy it will need to provide an interface with the security kernel that can only be invoked by trusted subjects. Trusted subjects are recognized by the security kernel through an internal identifier. The intent is to provide a capability that otherwise would not be permitted by the security policy built into the kernel. For example (assuming a Bell & LaPadula type security model), since the security policy does not allow an untrusted subject to lower an access class of information that was over classified, this functionality could be performed by a trusted subject. Trusted subjects can be implemented as asynchronous processes, called trusted processes, or as extensions of the security kernel in which case they would be called trusted functions. The combination of the security kernel and all trusted software is referred to as the TCB. The entire TCB must be subjected to the same development techniques as the kernel if security policy is to be maintained with a high degree of assurance on the system.

An interesting question is whether secure CAIS implementation could be improved by having a trusted "CAIS kernel" within the CAIS implementation, but outside the TCB kernel.

5.1.5 Performance Considerations

In an attempt to achieve a small and simple kernel, a significant performance penalty may be paid. Stringent performance requirements make it difficult to develop a straightforward, secure design and implementation. The performance issue associated with the development of a kernel-based operating system has always been a major concern. Previous attempts at providing reference monitor functions in software resulted in reduced execution speeds. However, recent attempts to building kernel-based systems have shown that performance enhancements can be realized. Probably the most significant means to achieve adequate performance is to rely on considerable hardware support. The richness of the hardware that must be supported will have a considerable effect on performance. At an extreme, all kernel functions could be implemented as hardware instructions, where all the hardware architecture would be completely responsible for security. As with the supervisor/kernel trade-off (with respect to functionality), specific hardware trade-offs need to be considered: complexity, size, and performance. The hardware features and software mechanisms necessary for a kernel based operating system to perform adequately are achievable, albeit sophisticated. The specific hardware features desirable for kernel based operating systems are provided in many modern computer architectures. Although several past security kernel implementations have resulted in significant performance degradations due to inadequate hardware, there is no reason that a kernel-based operating system should perform any worse than a non-kernel-based system with similar capabilities.

5.1.6 Existing Trusted Products for Prototypes

In determining what B3/A1 products are currently available that could be appropriate for the CAIS application, we turned to the NCSC's Evaluated Products List (EPL). The EPL is a composite list of off-the-shelf commercial products and support systems that have been evaluated to be in compliance with a particular class of the criteria. It should be noted that the EPL report is independent from any consideration of overall system performance, potential applications, or particular processing environment.

Currently, the EPL contains only one product at the B3/A1 level of trust, the Secure Communications Processor (SCOMP), STOP Release 2.1. The SCOMP has been determined to meet or exceed each of the requirements of the A1 evaluation class from the criteria. Besides the SCOMP, there is one other general purpose operating system that has been placed on the EPL--the Honeywell Multics, MR11.0, which has been evaluated at the B2 level. This product includes all the essential elements (features and assurances) for a B3 or even an A1 system. This means the Honeywell Multics, MR11.0 product implemented mechanisms are sufficient to capture the essential elements of a B3 product such that it could be used as a base for a secure CAIS prototype, as could the SCOMP, which has a B3 capability.

5.1.7 Conclusion

Computer security issues as they pertain to general purpose operating systems like the one that will be provided for CAIS 2 are very real. Methods of dealing with them are not always straightforward, and often performance and complexity trade-offs need to be considered. These trade-offs are in respect to the B3 security requirement for the kernel's requirement to be engineered as small and simple as possible, and the overall operating systems performance and complexity. From a pure security view point the kernel should only implement functions necessary to demonstrate compliance to CAIS security policy and the subsequent security policy model. Consequently, functionality included in the operating system to enhance performance or increase the convenience of writing software are not included at the lowest level occupied by the kernel.

Although it is difficult in the absence of a specific security policy to determine exactly what functions and services are needed within the security kernel, general kernel functions can intuitively be described. Briefly, the security kernel implements all functions with respect to resource management, process scheduling, memory management, and auditing. The security kernel must also function as the software portion of the reference monitor implementation in controlling access to objects in accordance with the security policy.

A major aspect of a kernel-based operating system is the means in which I/O is handled. I/O can be performed in two ways. The first, referred to as programmed I/O, requires a program controlled by the CPU to execute an I/O instruction.

This results in the transfer of information between the I/O and memory. In this case, the kernel would view the I/O device as an object and would control it as it would another object. The second method provides independent I/O processors that, once activated, asynchronously transfer information between device and memory. With this method the kernel would view the independent I/O device as a subject (active entity). As in programmed I/O, a the kernel would need to control the initiation of the I/O process through limiting execution of start I/O request to its domain.

The NCSC's most recent EPL reveals two possible candidate systems for secure prototypes: the SCOMP STOP Release 2.1 and the newest arrival to the list, Honeywell Multics, MR11.0. The SCOMP was evaluated by NCSC to be in compliance with the A1 class of the criteria, while the Multics was evaluated at the B2 level. Although Multics has not achieved at least a B3 rating, it does fall into the B3 range of feasibility, meaning it has the essential elements of a B3 product.

5.2 SECURITY IMPLICATIONS OF DISTRIBUTION

This section describes the issues of security for a distributed CAIS environment. It focuses on the security aspects of CAIS 2, but avoids any details regarding CAIS, per se. Important issues addressed are the requirements for achieving a B3-certifiable, distributed CAIS with respect to the anticipated trusted network evaluation criteria. Many of the ideas presented here are based on research reported in the group working papers published in an invitational workshop on network security [6].

The Trusted Computer System Evaluation Criteria (TCSEC) (CSC-STD-001-83) sets forth security certification criteria for a so-called monolithic computer system. It specifically avoids security issues outside the domain of single, integrated computer hardware and software systems. Thus, the orange book criteria do not sufficiently answer the many questions raised when considering a secure network. Therefore, the existing criteria found in the TCSEC can, at best, be considered the baseline for creating new criteria with which network systems may be evaluated.

Several questions arise regarding security and networks. This section lists pertinent questions organized under headings found in the TCSEC.

5.2.1 Definition of Terms

It is important to define just what is meant by several terms before going on. First, what is meant by a "network." Perhaps the most significant development is the trend towards distributed systems. Generally, distributed systems are composed of smaller systems communicating over some medium to achieve a stated, common goal. Systems housed in a single "box" that comprise multiple processors communicating over a private bus or local area network must be included in an example of a network.

Trusted System. One which employs sufficient hardware and software integrity measures to allow its use for processing simultaneously a range of sensitive or classified information.

Trusted Computing Base (TCB). All the protection mechanisms within a network that enforce a security policy on that network.

Network Subjects.

TBD

Network Objects.

TBD

5.2.2 Questions Unique to Networks

Can one policy cover the entire network? Is the Bell and Lapadula model in the current TCSEC appropriate as a network security model? What are "subjects" and "objects" in a network? How can possible heterogeneous security models on different nodes of a network be connected under a uniform policy and model? What are the host-to-host policy interfacing requirements? Can hosts that are certified at different levels (e.g., A1 and C2) be interconnected? Should the CAIS design address a distributed secure implementation?

5.2.2.1 Access Control Questions

Is discretionary access control (DSC) possible in a network environment? If so, how might one implement it across the network (i.e., inter-host access)? What granularity of labeling and permissions are needed? What mandatory access control (MAC) policy should the network enforce on a network-wide basis? How is information separation accomplished in a network? Where might mandatory access controls be implemented--on one machine, distributed, etc.?

5.2.2.2 Accountability Questions

Where should identification and authentication be done? What granularity is required: users, hosts, etc.? Where should auditing be done? Should each node maintain a separate audit? What does the trusted path concept mean in a network environment?

5.2.2.3 Assurance Questions

What is the role of a TCB in a network environment? Could the duties of the TCB be distributed? Should they be? Is it possible to maintain minimal TCB complexity in a network? What role do protocols play in protection of sensitivity labels and data being passed along the network?

5.2.2.4 CAIS Boundaries

What are the security requirements for one CAIS implementation which controls a distributed system as a whole? Which of these require interfaces visible at the CAIS level?

What are the requirements for intercommunicating between secure CAIS implementations? Can a secure implementation export information to a non-secure implementation? What CAIS interfaces are needed? Are these tool interfaces?

5.2.3 Network Security Policy

The following sections describe the network secrecy and integrity policies.

5.2.3.1 Secrecy Policy

When considering secrecy policy and models, systems of systems (i.e., networks) differ from monolithic systems in two important ways: (1) component systems are active entities, and (2) component systems are autonomous entities. These two aspects impact security in that the "old" rules regarding reading and writing apply to passive "objects" in monolithic systems. However, network systems contain "readers" and "writers", sometimes called "file servers," that request information be read or written. Since mutual cooperation between requester and server must exist, once benign requester and request fulfiller (cooperating tasks) can compromise security. The basic security conditions fundamental to protecting from compromise no longer hold. The next paragraphs explain why.

The simple security condition states that in monolithic systems, a subject may read an object if security clearance of the subject "dominates" the security classification of the object. If, for example, a Top Secret subject requests to read a file classified Secret, the simple security policy permits the read operations. Similarly, the policy disallows a Secret subject from reading a Top Secret file. In a network, the simple security condition does not apply unless special precautions are taken.

When a network subject on one host requests a read operation for data contained on another host, the subject sends a "read request" to the second host. The host "object" server must interpret the request, obtain the data, and return it in a message to the requesting system. Clearly, synchronization is involved between the subject requesting process on one host and the object serving process on another. Because both subject and object are active, intelligent processes, a covert channel can be established in which the high level subject may encode information in the read request sent to the lower level system. The lower level system may, at a later time, save the now-decoded covert information at a lower level of classification.

A similar situation occurs when information is written from one system to another. The Bell and LaPadula *-property (star property) states that an untrusted subject may write (append) information into an object at a higher level. Again, networks using requesters and servers can compromise classified information. Because the receiver (object) of information must acknowledge receipt of the information, a signalling channel can be established via the acknowledgement message sent from the high level receiver to the lower level requester. The problem, in this case, is one of autonomy of the sender and receiver.

In the case of reading and writing, the root problem is the required end-to-end acknowledgements required by the network system. High level processes can choose to modulate responses by encoding high level information in the response. To counter this potential security breach, both "ends" of the information transfer process (i.e., requesters and servers) must be trusted. When they have been shown to not compromise security, they can be used in a network system safely and will enforce both the simple security property and the *-property. Another solution is to use an intermediary at the same level as the sender (or receiver). The remote intermediary at the secret level can receive information from a secret sender. The remote intermediary can then "up write" information at the Top Secret level within one host.

5.2.3.2 Integrity Policy

We have said that a network shall be protected from secrecy violations. To accomplish this, we utilize the concept of labeled information to facilitate the secrecy mediation process. Once we label information, we need to establish a protection against modification of the information contained within those labels. We need to protect their integrity. Once we have established the need for integrity, we are logically led to the need for a network integrity criteria. This must be an integral part of the network security policy. The integrity portion thereof, would state the criteria of data correctness. Some users would have more right to change objects than others. Thus, we have a potential hierarchy of integrity preserving behavior not unlike the criteria for secrecy. It can be shown that integrity based information changes have a directional flow which complements that of secrecy. Disposition of data is governed on the basis of security classification and need-to-know, whereas a need-to-modify underlies the concept of integrity. An integrity threat (sabotage) may be defined as an unauthorized modification of data. The protection policy must respond to each potential threat. A network is said to possess the property of integrity if it can be trusted to adhere to a well defined code of behavior. That code of behavior will have been predetermined by the network designers and the later concern is thus the guarantee that the network will perform as it was intended to perform by its designers.

5.2.3.2.1 Multi-Level Integrity

Desiring to effect control and restrictions on the way information may be passed from task to task, relative to integrity preserving implications, we are led to the concept of multi-level integrity. By ordering these levels in a determined hierarchy, we can then stipulate by construction that the "less important" applications cannot interface with the operation of the "more critical" applications. More importantly, objects would be inaccessible to subjects of lower integrity level attributes. The levels are based on information content and the modifiability thereof. Each process, which is composed of subjects, is assigned an integrity level equal to the integrity level of its component subjects. Each process is associated with a user. Thus, the protection properties of a process should be derived from those of its users. Individual objects and objects organized into files and directories would likewise be assigned integrity levels. Thus, we have multi-level integritized objects. Files, directories and nodes would then also have to be hierarchically ordered in terms of integrity level attributes.

The set of integrity classes is disjoint from the set of security classes, but is analogous. A representative set may be defined as:

Integrity Level	Analogous to Secrecy Level:
Crucial	Top Secret
Critical	Secret
Important	Confidential
Ordinary	Unclassified

An integrity level for a computer network element is composed of an integrity level, which relates to and identifies the importance of the element, and, additionally, a set of integrity compartments identifying the information partitions the element may contain or access. These compartments are assigned on a basis which ultimately rests on the trustworthiness of individuals. Similar considerations apply in the assignments of integrity levels for individuals as they do for assignment or security levels. A subject is assigned an integrity level commensurate with the level of its user and with the principle of least privilege; that is, tasks will be confined to the least privilege necessary to effectively carry out the task successfully, legitimately and in accordance with policy.

5.2.3.2.2 Integrity Level Compartments

Just as security is structured for compartmentalized information disclosure, integrity is similarly structured for compartmentalized information modification. The need to compartmentalize data modification may manifest itself in a variety of ways. Some integrity compartments might identify differing applications areas, logistics information, real-time C2, disparate commands, etc. Thus, in order to qualify as a purveyor of a proper modification, one would need to possess access rights to the appropriate compartment of a target object in order to effect the modification. In other words, there would need to be a requisite "need-to-modify". Compartments, thus, increase the range of level assignment flexibility and accurance by focusing on the "need-to-modify". Compartments can also be looked upon as a network performance maintenance device. A compartmentalized object is stating to other elements that certain characteristics must be maintained when handling the object.

Finally, justification for compartments may lie in the flexibility they give in the control of issues affecting the other aspects of a secure network; that is, secrecy and necessity. Because of threat overlap from one aspect of a secure network to another, integrity level compartments serve as a further delimiter and control device in structuring the network and managing its overall security.

Because of the nature of the compartments and their connection to a user requirement of "need-to-modify", a technique of "Level Override" may be possible. That is, if a subject is not in possession of a "need-to-modify" a particular object, it may be possible to bar the modification even if the subject possesses a superior integrity level attribute. For example, (on the user level) an individual whose compartment set contains only

ATOMIC cannot pass information to an individual whose compartment set does not contain ATOMIC, independent of the former's integrity level or the other components of the latter's compartment set. Network design considerations of this sort would always involve network specific policy related decisions.

5.2.3.2.3 Network Performance Ramifications

A major problem area in developing a secure network is the integration of secrecy, integrity, and necessity constraints and the potential combinative volatility of their interaction. First, we must independently identify the three separate mechanisms and classes of policies which are to be supported and then discuss the interactions amongst those policies.

For example, once we have discussed secrecy attributes and realize that information in such a model does, in fact, flow "upward", we see that such a multi-level security model does not p·ohibit a process at some security level from modifying information at a higher security level. However, such a prohibition may be desirable and might be the intent of a given integrity policy. This, integrity criteria may superimpose a delimiting effect on modifiability. That is, a person may have Top Secret (Atomic), but no need to modify a Secret (MATO). As a practical example, a person might have possessions of a document through a need-to-know at Top Secret, but have no commensurate right to make a change in that document (no need-to-modify) at any clearance level. This, "need-to-modify" (and integrity designation) would delimit a person's right to access or, at least, confine his mode of access greatly. Integrity compartments (through which need-to-modify may be expressed), although non-hierarchical and, thus, equal in level but different in content, can be used to further delimit integrity levels and, by extension, their interaction with secrecy and necessity. An example of an integrity implication on a necessity attribute would be the effect of the incremental change in a priority timer which falls outside the permitted integrity preserving "range of correctness" for that timer. If this were the case, denial of service would possibly come into operation, or, at least, delay of service, which, depending on the circumstance, could result in the same system effect. Thus, we could have a process which adheres to secrecy tenets, but, nevertheless, would create an integrity violation which has affected necessity and finally has resulted in erroneous network performance. It is easy to see that there are countless interactions of this sort, all of which must be accounted for and prohibited or allowed depending upon networks design and proof of that design's adherence to validity.

5.2.3.3 Necessity

TBD

5.2.4 Network Security Model

A secure network model can use processing nodes or entities that have been evaluated using the Orange Book criteria to preserve the security provided by separate TCBs through a set of interconnection rules. The composite network, termed a trusted network, utilizes TCB technology to the greatest possible extent and builds on previous research. (see "2-10" paper)

If users could function in total isolation of each other, security policy for such a system could be one of complete separation. Clearly, however, users must share information with one another and invoke processes on other users' data files stored at various sensitivity levels. In trusted computer systems, sharing information is controlled by security policy models. The Bell and LaPadula model, for example, defines the allowed accesses subjects have to objects. However, this model is an example of one that addresses only secrecy (compromise), and only for monolithic systems. Two other security issues must be incorporated into any proposed model to yield a secure network: an integrity model and an availability (also called delay/denial or necessity) model. The three issues of secrecy, integrity, and availability collectively form the network model. These are discussed in the sections that follow.

5.2.4.1 Secrecy Model

TBD

5.2.·.2 Integrity Model

TBD

5.2.4.3 Necessity Model

TBD

5.2.5 General Network Security Considerations for Certification

The following sections describe general network-wide questions, access controls, accountability, and verification and covert channel analysis.

5.2.5.1 General Network-wide Questions

TBD

5.2.5.2 Access Controls

TBD

5.2.5.3 Accountability

TBD

5.2.5.4 Verification and Covert Channel Analysis

TBD

5.2.6 Network Security Issues and CAIS

TBD

5.2.7 Summary and Conclusions

TBD

5.3 ACCESS CONTROL ON ALL NODFS VERSUS ON LEAVES

Lacking a security poli:y or any other specific guidance as to the intent or requirements for security in CAIS, we shall address some of the aspects of CAIS design which affect access control. In ·eneral, we will address discretionary access control, some perceived deficiencies, and some recommendations.

One question concerns whether access control should be applied on every node in a path or just the end. The primary impact of this issue is in traversing nodes in a path to a target node. Conventional theory on access control deals with the relationship between a source subject (user or process) and a target object (file or data item). The node structure of CAIS which introduces the possibility of an extended path to both subject and object complicates the issue. This means that any access will nearly always involve traversing a number of nodes to reach either subject or object.

Since performance is of significant concern with CAIS 1, it seems clear that anything more than minimal access control checks at each node in a path for each access would be prohibitively costly.

5.3.1 Adopting a Role

Roles are defined as intermediate structural nodes to which access control attributes may be assigned (for example a group node), thus enabling a given process node (or nodes) to assume more than one access control configuration.

5.3.2 Setting Classification Level

There is some concern relative to how an object's classification level is set. Section 4.4.3.2 of CAIS 1 correctly states that labels shall reflect the security level of user, but only for the root process node. It appears to say that non-root node labels can be assigned by the creator (as well as set by default from parent node). In a multi-level secure system created data must reflect the classification level of originator and provision must be made to allow user to log-in at various classification levels. The user will only be operating at one level at a time. Access control mechanisms will allow user to write-up and read-down relative to level he is operating at, at the time.

All processes and data created by that user should be labeled by the system at the level under which the user is logged in. If he wishes to create products at another level he must log out and log back in at that level. This insures that products he create have the proper sensitivity label and that the access control system can properly control them. Allowing the user to designate classification levels leaves the system open to abuse and subversion.

It is unclear if the mechanism defined in CAIS 1 can accentuate leaks due to covert :hannels. An additional troublesome point is that in the CAIS 1 model, a process is both a subject and an object. A question .s whether this also opens opportunities for abuse. (See 5.2.3.1 :below.)

An additional point to note on sys:em operation is that the levels which a user is allowed to log in under should be controlled by the system/security supervisor logged in on a protected and trusted process. This ensures that there is only one control point for subject/object label assignments.

5.3.3 Problems with CAIS 1 Discretionary Access Control Mechanisms

Discretionary access is based on the concept of "need-to-know." That is, given that the subject possesses the mandated level of clearance to access an object which is marked at a given level of sensitivity, his access is further restricted based on whether he really needs the information to do his job. An example might be a contract specification. The designer needs to know about the structural details but not about the cost figures involved. Or he may need to know about the costs pertaining to his portion of the project but not the rest. His access to that information is based on the discretion of his supervisor.

It is important to recognize that the role of discretionary access is in addition to and within the constraints of mandatory access control. It is an added-on capability which is nice to have, as it gives the owner of a data item or process the power of selective access control subject to the satisfaction of mandatory access requirements. Security requirements can be met without discretionary access control at all. <u>In fact, no discretionary access control scheme has been shown to be provably secure in a multi-level system.</u> When a system is operating with multiple levels of sensitivity or classification, discretionary access must be considered only if the requirements of the mandatory access control policy are satisfied.

The discretionary access control mechanism is incompatible with the mandatory access control mechanism. Since CAIS is an interface description it does not necessarily imply that the use of mandatory control is precluded by the use of discretionary controls. The invocation of access control is a operational function, not an interface one. Conceivably access could be made contingent on the satisfaction of both mandatory and discretionary checks. However, there may be a performance penalty involved since two separate mechanisms would be required:

one to check object access relationship grant attributes AND subject role relaticnships for discretionary checks. Both must be checked and satisfied if both are in use.

It would be more efficient to put discretionary access information on the same label with classification and compartment information. This would facilitate the use of ACLs as suggested by the Orange Book. Access Control Lists are labels attached to objects which provide the sensitivity and compartment information required by mandatory access rules, and a list of authorized users (in CAIS - nodes) along with their respective access rights for discretionary access. The ACL concept should be compatible with the grant attribute of the access relation. Instead of a separate "grant" process the same model would be applied to both mandatory and discretionary parts of the label. This should be more efficient and better meet Orange Book requirements for a B3 security rating.

5.3.4 Recommendations

It is recommended that the current discretionary access control mechanism be replaced with an access control list mechanism which will expand the current label to include discretionary access rights for each listed node.

Further it is strongly recommended that a package be created to address the user log-on function which, under the control of the system/security supervisor, will allow a user to log-on at various assigned classification levels. The level of the user root process node will be used from then on to label created object nodes.

ISSUE SUMMARY

ISSUE REPORT REVISION HISTORY

ISSUE 6.0 -- Analysis of Performance Tradeoffs

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	6/30		First Draft

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

MAJOR ISSUE: Is the aggregate CAIS design implementable with acceptable cost and performance?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
6.0	Analysis of Performance Tradeoffs	
6.0/0	OPEN	What are the expensive CAIS features? How expensive are they?
6.0/0	OPEN	Given the aggregate CAIS design, is there a reasonable expectation that an acceptably performing implementation is possible?

CHAPTER 6

ANALYSIS OF PERFORMANCE TRADEOFFS

The following sections describe performance tradeoffs in CAIS 2.

6.1 INTRODUCTION

Performance is the degree to which an operation is efficient. This efficiency may be characterized in terms of speed of the operation or in terms of utilization of resources such as processors, memory, disk storage, communication bandwidth (DoD Requirements and Design Criteria for the Common APSE Interface Set (RAC), section 2.3). This requirements document [8] makes reference to performance considerations and to the objective of ultimate system usability. System performance affects system usahility, most notably via job throughput and perceived user response time. The complexity inherent in the implementation of an environment such as the CAIS jeopardizes system performance. Although each feature examined individually seems to have an acceptable cost, the aggregate design may not perform acceptably. The intent of this section of the Issue Report is to highlight and summarize performance issu s of the intended CAIS 2 facilities as an attempt to characterize and avoid aggregate performance problems.

This report will consider and discuss the performance characteristics of various facilities named in the CAIS 1 Requirements [8]. Design features defined and/or implied by these facilities are taken from the RAC/CAIS 1 Comparison and Analysis [7]. Note that in the absence of a specific design, it is difficult to anticipate performance accurately; these performance characteristics will therefore be approximate. The most efficient systems will depend on high performance in their most frequent operations. This report will present only those features expected to require operations of impact to overall performance.

6.2 PERFORMANCE EVALUATION STRATEGY

Performance of selected CAIS features will be evaluated in terms of the primitive operations required by each. These operations include the following as a minimum:

a. Process startup (and termination),

b. CAIS calls,

c. I/O channel open (and close),

d. Relationship traversal,

e. Type check(s),

f. Trigger check,

g. Access check,

h. Attribute examination,

i. Audit record posting,

j. UID generation and posting,

k. Reading,

l. Writing,

m. Storage consumption,

n. Concurrency control (locking),

o. Waiting and,

p. Schema modification.

The remainder of this chapter will focus on the strategy for evaluating CAIS performance. The CAIS 1 prototype and the Ada Language System (ALS) will provide the starting point.

The order of presentation follows that of the RAC.

6.3 PERFORMANCE ISSUES

The following sections describe the performance issues of specific RAC2 requirements.

6.3.1 implementability

Implementability as specified (RAC2.3) requires configurability [7]. Configurability is the ability to assemble reduced, or degenerate, forms of some CAIS components. One possibility is for users to be able to configure CAIS implementations statically at the time the Kernel Ada Programming Support Environment (KAPSE) is installed. Alternatively, users could configure the implementation dynamically at the initiaton of a session. An example of a degenerate configuration would be a nonsecure CAIS installation. In this example, as in others, degenerate functionality would not affect total process startups and terminations, CAIS calls, or I/O channel openings and closings. Calls to set or check security information would still be made, yet these calls would be satisfied with a quick return. CAIS performance would improve with this decrease in actual operations. The primitive operation that would be eliminated in this example is access checking. Depending on the nature of the specific degenerate configuration, other primitive operations would be involved. This potential for performance improvement exists whether the degenerate configuration is established statically or dynamically.

6.3.2 Security

Security as specified (RAC2.8) necessitates a security policy defining both mandatory and discretionary access control. Access rights and rules must be established and stored. Potentially, The necessary checking of access rights against access rules will slow all operations, including process startup and termination, CAIS calls, I/O channel opening, relationship traversals, attribute examinations, reading, and writing. Access checking will be discussed in section 6.3.11.

The primitive operations relating to security are:

a. access checks, and

b. storage consumption.

6.3.3 Exceptions

RAC3.2C suggests the need for a mechanism to pass exception information beyond that supported by the Ada exception handling mechanism [7]. At a minimum, a mechanism to relate exceptions to specific executing tasks is needed. Assuming the existence of unique task identifiers, a table could be used to record the most recent exception for each task. This solution would affect performance because of the necessary storage, access, and logging overhead of maintaining such a table. A more complex related issue is the necessary mechanism to maintain task information in general. This task-specific information would include a unique identifier, the owning process, and the active lifetime. CAIS calls would suffer the overhead of passing this additional information.

The primitive operations affected are:

a. Process startup and termination,

b. CAIS calls, and

c. storage consumption.

6.3.4 Entity Management Support

Entity Management Support (RAC4) implies a meta-data-description method such as a typing mechanism, a schema-subschema mechanism, or both. Since users may install tools at their own discretion, this mechanism (the schema) must be modifiable. Schema modification would create a "concurrency bottleneck" by requiring exclusive access to this system-wide resource, and this bottleneck would degrade the performance of competing concurrent processes.

The Entity Management Support specification also implies support for relationships. Evaluation of the performance of relationship support depends, in part, upon how many relationships exist in a working system and how frequently these relationships are traversed. Performance will suffer when the frequency of traversal is high and the relationships are not localized. Alternatively, "anticipatory caching" of soon-to-be accessed relationships could alleviate this performance degradation.

Entity Management demands name space control of nodes, attributes, and relationships, in each case to prevent tools from using the same type name for different purposes. Performance of type definitions is most sensitive to the design of the name space control mechanism. As the type name space becomes larger, storage of the names and mapping to the actual locations increases as a performance issue. However, hashing would offer improved performance independent of the size of the name space.

Entity Management further implies a typing mechanism as a foundation for name control, to describe the representation of the data, and to assert and validate legal operations on the data. Performance of every CAIS operation is subject to the typing mechanism imposed.

The primitive operations involved in entity management are:

a. process startup and termination,

b. CAIS calls,

c. opening/closing an I/O channel,

d. relationship traversal,

e. type check(s),

f. trigger check,

g. access check,

h. attribute examination,

i. post an audit record,

j. generate and post a UID,

k. reading,

l. writing,

m. storage consumption,

n. concurrency control (locking)

o. waiting, and

p. schema modification.

6.3.5 System Integrity

System Integrity, as specified (RAC4.1C), necessitates access control, a mechanism for database backup and restore, adequate mechanisms and controls for transactions, and the posting of an audit trail. Access and transaction control are themselves features specified in the CAIS Requirements, and will be discussed in sections 6.3.11 and 6.3.12.

Database backup and restore are nontrivial when attempting to back up each node exactly once within a tree structured system such as the CAIS. Performance is particularly sensitive to the relationship traversal time, degrading with the number of nodes, and with the number of repeated visits to each node. A partition of the nodes would help to reduce the inefficiency of determining repeatedly if each node has yet been "backed up". The primary/secondary relationship scheme, if maintained, would provide one possible partition. Internal orderings should also be considered, as should be nonpartition schemes in which nodes are marked (perhaps with a timestamp) as they are saved. Internal orderings and markings require additional storage, and slow performance slightly with the added step of accessing the ordering or marking prior to accessing the node to be backed up.

The primitive operations pertaining to system integrity are:

a. CAIS calls,

b. opening an I/O channel,

c. relationship traversal,

d. access check,

e. type check (s),

f. trigger check,

g. attribute examination,

h. reading,

i. writing,

j. concurrency control,

k. posting an audit record, and

l. storage consumption.

6.3.6 Rules About Type Definitions

Rules about Type Definitions (RAC4.2B) may manifest various levels of complexity. For example, a definition of a relationship type might state that only "one to one", "one to many", "many to one" or "many to many" instances of that relationship type are to exist between node types. If so defined, this rule would have to be represented and enforced. Enforcement of such a rule would require rule checking upon every relationship creation and modification. The amount of checking increases with the complexity of the type definition rules. Performance slows with increased checking.

The primitive operations of interest are:

a. CAIS calls,

b. relationship traversal,

c. type check(s),

d. trigger check,

e. access check,

f. post an audit record,

g. reading,

h. writing,

i. storage consumption, and

j. schema modification.

6.3.7 Changing Type Definitions

Changing Type Definitions (RAC4.2D) has varied performance implications depending on the interpretation. For example, if a type may be changed or deleted while instances of that type remain in the database, then perhaps all such instances must be removed. This particular example would degrade the performance of competing concurrent processes which would be forced to wait during (exclusive) system-wide database cleanup.

The primitive operations involved are:

a. CAIS calls,

b. opening an I/O channel,

c. relationship traversal,

d. type check(s),

e. trigger check,

f. access check,

g. post an audit record,

h. post a UID,

i. schema modification,

j. concurrency control, and

k. waiting.

6.3.8 Triggering

Triggering (RAC4.2E) is a mechanism whereby prespecified procedures or operations are automatically invoked upon the occurrence of specific events. Providing for triggering implies that every time any "event" occurs, an internal list of "triggers" must be examined for possible applicability. This feature would affect every CAIS call, every process startup, every relationship traversal, attribute examination, read, and write; in short, every operation. Triggering is therefore viewed as causing an unavoidable degradation in performance.

The primitive operations involved are:

a. process startup and termination,

b. CAIS calls,

c. opening/closing an I/O channel,

d. trigger check,

e. type check(s),

f. access check,

g. post a UID,

h. post an audit record,

i. concurrency control (locking),

j. waiting,

k. storage consumption,

l. read,

m. write, and

n. schema modification.

5.3.9 Identification Methods

Identification Methods (RAC4.3C) must provide exact identification of all entities and relationships. An identifier to be assigned to every entity and relationship could be derived from information about the object's originating organization, the AL3 database identifier, and the object's serial number. To accommodate the necessary universal uniqueness of this identifier, approximately 40 bits of information must be maintained. The performance of storage, reading, writing, and relationship traversal would be degraded. Designs such as anticipatory caching to speed up relationship traversal would be prohibited, as the number of likely relationships (hits) in an arbitrary block of memory would decrease enormously as a result of all the stored identification information.

The primitive operations involved are:

a. process startup,

b. CAIS calls,

c. opening an I/O channel,

d. relationship traversal,

e. access check,

f. trigger check,

g. post a UID,

h. post an audit record,

i. storage consumption,

j. read, and

k. write.

6.3.10 Synchronization of Operations

Synchronization (RAC4.4F) mandates dynamic access synchronization mechanisms for individual entities, relationships and attributes. This requires concurrency control, i.e., locking mechanisms for every existing entity, relationship, and attribute. Locks must be created, stored, set, checked, released and removed. The storage and time used for locking depends on how the locking mechanism is implemented, although lock checking must add overhead to every operation attempting to access any entity, relationship, or attribute. In general, the most significant performance impact of locks is the waiting that ensues when set locks cause access denial.

The primitive operations of interest to synchronization of operations are:

a. CAIS calls,

b. opening/closing an I/O channel,

c. relationship traversal,

d. attribute examination,

e. trigger check,

f. post an audit record,

g. reading,

h. writing,

i. concurrency control, and

j. waiting.

6.3.11 Access Control

Access Control (RAC4.4G) demands selective prohibition of operations on entities, relationships, and attributes as requested by individuals. Performance is subject to the efficiency of checking defined access rights against defined access rules during process startup, I/O channel opening, relationship traversal, and attribute examination. Since access codes may be modified, the performance impact of access control depends additionally upon how often such modification may occur. If, for example, access rights may be modified on a file already open for I/O, then the file's access privileges must be checked on every read/write operation, rather than just once at the time of file opening. In general, the frequency of such checking operations increases with the number of occasions on which access modification is permitted. Performance degrades with the frequency of necessary checking. Modification in general impedes performance, as exclusive control of the object to be modified is always required.

The primitive operations of interest are:

a. process startup,

b. CAIS calls,

c. opening an I/O channel,

d. attribute examination,

e. relationship traversal,

f. trigger check,

g. access check,

h. reading,

i. writing, and

j. post an audit record.

6.3.12 Transaction Mechanism and Transaction Control

Transaction Mechanism and Transaction Control (RAC4.5A,4.5B) protect the integrity of global data subject to modification by individual and/or concurrent processes. The necessary ability to "back out" or nullify the effects of any incomplete or failing transaction requires either that all such effects are held in suspension until transaction "completion" (and then enacted "all-at-once"), or that a journal mechanism support any desired "backing-out". Each approach manifests its own drawbacks and performance characteristics.

If transactions are held in suspension until "completion", all temporary effects must be stored. The final set of effects must be committed to the database as one indivisible action. At this time, exclusive control of the database must be maintained, locking out all competing concurrent processes. Performance of all processes wishing to access the database will degrade in direct relation to the time needed to commit the completed transaction's effects to the database. This time may be controlled, in part, by a limitation on the allowable size of a single transaction.

Journal mechanisms avoid the exclusive control performance problem associated with suspension mechanisms, but require sophisticated interweaving of the transactions (and effects) of multiple, possibly distributed, users. Storage of the journal is a performance concern in that every journal entry might require a (slow) write to disk. This performance degradation may be lessened by careful selection of what is recorded in the journal.

The primitive operations called for are:

a. trigger check,

b. post an audit record,

c. concurrency control,

d. waiting,

e. reading,

f. writing, and

g. storage consumption.

6.3.13 History Mechanism

History Mechanism (RAC4.6A) necessitates history generation and retention. The additional task of recording every operation affecting entity, attribute, and relationship values in the history log degrades performance in a minor yet universal manner. Each operation suffers the overhead of the logging, and logging itself may require a time consuming write to disk. Storage may be degraded as long sessions result in long history logs.

The primitive operations involved are:

a. CAIS calls,

b. opening an I/O channel, and closing it,

c. attribute examination,

d. post an audit record,

e. generate and post a UID,

f. reading,

g. writing,

h. storage consumption, and

i. schema modification.

6.3.14 Synchronization of Program Execution

Synchronization (RAC5.4) requires task waiting, parallel execution of processes, the coordination of cooperating processes, and means for process suspension and resumption. Task waiting and explicit process suspension clearly penalize performance. The remaining synchronization requirements call for locking mechanisms at the process and task levels. Similar to those for the synchronization of operations, these locks must be created, stored, set, released, checked and removed. Again, the most significant performance impact of locks is the waiting that ensues when set locks cause access denial.

The primitive operations involved in execution synchronization are:

a. process startup and termination,

b. CAIS calls,

c. trigger check,

d. reading,

e. writing,

f. storage consumption,

g. concurrency control, and

h. waiting.

6.3.15 Input/Output

Input/Output (RAC6) requires logical device drivers, standard text representation, standard graphical representation, standard data interchange descriptors, and window management [7]. Each of these features represents a level of decoupling between tools and the processes which use them, a decoupling aimed at improving portability. Each such additional level degrades performance however, by introducing the overhead of interprocess communication or of procedural interfaces.

The primitive operations of interest are:

a. process startup,

b. opening/closing an I/O channel,

c. trigger check,

d. access check,

e. post an audit record,

f. reading, and

g. writing.

6.3.16 Input/Ouput Sequencing

Input/Output Sequencing (RAC6.3H) demands user dialogue control [7]. Since the CAIS must support non-windowed devices, terminal locking will be required. For example, if several processes are using one terminal and one of these is waiting for a response from the user, the other processes must be locked from the terminal until the response is satisfied. In such a case the performance of the locked-out concurrent processes will most likely be severely degraded.

The primitive operations involved are:

a. trigger check,

b. concurrency control, and

c. waiting.

ISSUE REPORT REVISION HISTORY

ISSUE REPORT REVISION HISTORY

ISSUE 7.0 -- Taxonomy of Distribution

REV NUMBER*	DATE	SECTION REVISED*	DESCRIPTION OF REVISION
0	6/30		Shell only.
1	7/29		First draft, change of title.

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

ISSUE REPORT REVISION HISTORY

MAJOR ISSUE: What tool interfaces are needed to support distributed CAIS's?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
7.0 Taxonomy of Distribution		
7.0/0	OPEN	In what ways is the CAIS likely to be distributed?
7.0/0	OPEN	Is there a difference between inter- and intra-CAIS operation under some distribution scenarios?
7.0/0	OPEN	What are the boundaries of a distributed CAIS?

CHAPTER 7

DISTRIBUTION OF CAIS IMPLEMENTATIONS

This issue report is divided into four sections. First, a taxonomy of distribution is offered. This taxonomy is used to provide a common basis for discussion. Next, the International Standards Organization's Reference Model of Open Systems Interconnection (OSI) is summarized. It will be argued that only a few layers of the OSI Model concern CAIS directly. Many of the traditional concerns of distribution are hidden by lower-level software and/or hardware. This section will discuss those topics which are not hidden, and thus must be dealt with by the CAIS interfaces. The third section will offer a definition for and discussion of the boundary of an instance of CAIS. A small subset of the distribution taxonomy will be used to demonstrate several typical CAIS distribution scenarios. Finally, a list of CAIS - specific distribution issues will be addressed.

7.1 DISTRIBUTION TAXONOMY

The following distribution taxonomy is adapted from "Computer Networks," by Andrew S. Tanenbaum, Prentice Hall, 1981 [9], and The Report on the ACM SIGOPS Workshop on Accomodating Heterogeneity by David Notkin, et al, Technical Report 86-02-01, Dept. of Computer Science, Univ. of Washington, March 1986 [10]. It gives a macroscopic view of distribution concerns, not all of which are directly applicable to CAIS, but which nevertheless establish context for CAIS issues.

Distribution can be viewed as containing three design spaces. These include computation, operating systems, and file systems. Most choices of computation, operating system, and file system support are mutually independent.

7.1.1 Computation

Five models of distributed computation are identified. These include:

a. Hierarchical Model

b. CPU Cache Model

c. User-Server Model

d. Pool Processor Model

e. Data Flow Model

Diagrams of each of the five models are included in Figures 7-1 through 7-7. A short discussion of each model follows.

7.1.1.1 Hierachical Model

This model forms a tree. A different level of computational detail is usually handled at each level of the tree. This model is often applied in large corporate data processing systems. In such a system the lowest level of the tree could handle factory floor numerical control. The next level may handle factory-wide inventory control, the highest level corporate finances. File transfer and electronic mail are the typical communication media. Remote execution of processes is seldom a concern in this model.

7.1.1.2 CPU Cache Model

In the CPU Cache Model the system workload may exceed the capacity of any single CPU. The decisions on where to perform computation is based on a combination of machine characteristics, computational costs, communication bandwidth, and current workload. Workload decisions are based on system state information rather than user choices. Such workload scheduling is often called the "assignment problem." Typically, programs run on the central CPU when it is available. The assignment problem is largely a static problem in this model. Analysis of the system and machine characteristics may predetermine the assignment of each job, or leave only minor run-time choices up to the system.

7.1.1.3 User-Server Model

With the introduction of more powerful mini- and microcomputers, the functional distinctions assumed for the role of satellite machines in the hierarchical and CPU-Cache models were no longer as clean-cut. In this model each user has his own machine (computational resource with or without some local secondary storage). A fast Local Area Network allows high-speed reliable communication among a group of machines, many of which perform specialized tasks as servers to the personal computational resources.

7.1.1.4 Pool Processor Model

The Pool Processor Model is similar to the User-Server Model, except that users do not have private computational power. Users interface to the system with only a terminal. Processors are allocated dynamically. This model must provide a scheduling algorithm. One common algorithm is called "bidding," where each processor bids on a process, or other primitive unit of work. The lowest bidder gets the job. Such algorithms must address issues of deadlock detection and/or prevention.

7.1.1.5 Data Flow Model

The Data Flow Model has no variables, program counter, or memory as are contained in "traditional" computers. Values are represented as packets transmitted between processing units. Each processor has a specific function. The results of each function are based only on the processor's inputs. There are no global variables or side effects. Computation may proceed as soon as inputs are ready. Parallelism is the primary benefit of such a system. Processor functions can often be selected from a template of allowable functions. The data flow model is not seen as being directly applicable to CAIS, but is included to complete the range of distribution models.

7.1.2 Operating Systems

The following sections describe different kinds of operating systems.

7.1.2.1 Network

A Network Operating System (NOS) is used to piggyback network services on top of an existing non-network system. Each host retains its original operating system. An NOS is generally easy to implement. It provides network access through an "agent." The agent may be the host or an independent network access machine. The agent may either make the network visible to the user via a command processor or converter, in the form of "At host X do command Y," or it may hide the network by an encapsulating technique which traps local system calls and converts them for use by the network.

7.1.2.2 Distributed

A Distributed Operating System (DOS) provides a single network-wide view of its computational and database resources. This is most often realized by either a global process model, in which processes are free to communicate with function calls or interprocess communication; or by an object model, where typing and capabilities (rights to access) objects are primary concerns. CAIS embodies both process and object concepts. Capabilities are functionally replaced by a combination of relationships and access rights.

7.1.3 File Systems (Adapted from Notkin paper)

Just as there are three primary independent design spaces in the distribution area, within the file system area there exist several dimensions to the design space. Andrew Black, of the University of Washington Deptartment of Computer Science, has made a comparison of file systems and arrived at eight design areas, seven of which are shown for representative contemporary file systems in Figure 7-6. A description of each of the design spaces is given below.

Accessibility refers to the view of the file system by individual users. The two extremes of this space are that all files may be accessible from everywhere (any processor), or all files must (at least logically) be retrieved from a central file server.

Access Transparency is another aspect of a user's view of the file system. The concern here is whether the host hides the difference between accessing the local and distributed file systems.

Location Independent Names are names which are independent of the physical location of the file. Names could be partially dependent on physical location.

Version Control refers to the techniques used to manage multiple versions of a single logical file if versions are allowed in the file system at all. Typical decisions are whether the version control is simply a naming convenience, built on top of unique files for each version, or whether some "change" information is retained from a common base to build subsequent versions. A balance is needed between storage efficiency and the speed with which a specific version can be recreated, especially if intermediate versions are (at least logically) deleted.

Read Only or Overwritable refers to file-system-wide choices as to whether all files are read only after the first time they are written, or whether they are overwritable. This differs from user-defined access controls on files which have no inherent system-imposed limitations. If files are overwritable, this design space includes the choice of locking mechanism to prevent conflicting writes.

File Typing is usually either strictly enforced or nonexistent. Attributing a type to a file may define its allowed operators and operations. This is used to avoid misuse of files such as running an Ada object image into a payroll program.

Replication refers to whether the system has any capability to maintain copies (presumably exact) of a file across distributed nodes.

Caching is the pre-reading of file data in anticipation of a request to access the data. Caching is done to increase throughput. The cache resides in a faster storage area than the file's physical storage medium. Storing a number of write requests and performing them all at once (write caching) is also possible. The parameters associated with caching, such as cache size, amount of data cached, and so on, are important considerations in the development of efficient programs. Some problems introduced by caching are how to handle updates to cached information if it is modified after it is cached but before it is requested by a program, and ensuring the availability of information in a write cache to potential readers.

Fetch Granularity refers to the portion of a file which is retrieved from a file server. Typically granularities are pages, variable size pages, entire file, and stream.

Some portions of this file system design space will be defined by CAIS, while others will be left up to the implementations. Table 7-1 shows which file system design areas will be specifically addressed by CAIS. A short discussion follows.

Accessibility -	IMPLEMENTATION DEFINED
Access Transparency-	CAIS
Location Ind. Names -	CAIS
Version Control -	CAIS
Read Only / Overwritable -	CAIS
File Typing -	CAIS
Replication -	CAIS
Caching -	IMPLEMENTATION DEFINED
Fetch Grain -	IMPLEMENTATION DEFINED

Table 7-1. File System Design Areas.

CAIS should place no restrictions on Accessibility, Caching, or Fetch granularity. Access Transparency will be available within a CAIS instance. CAIS will define a file naming convention. The naming convention will be location dependent, that is, dependent on the CAIS instance. Version naming will be defined by CAIS. CAIS files will be overwritable, although lower level access control can prohibit overwriting. Files will be typed. Interfaces for defining replicated files will be supplied. Replication among independent CAIS instances is not considered reasonable, as an import/export must take place in addition to other standard replication-tracking logic. If replication is desired among geographically distributed systems, they must be within the same logical CAIS instance.

7.2 OSI MODEL

A descriptions of the OSI Model is given, followed by a discussion of portions of the model applicable to CAIS. Figure 7-7 shows the ISO-OSI model.

7.2.1 Model Description

The physical layer is concerned with transmitting raw bits over a communication channel. The design issues at this level are largely mechanical, electrical and procedural interface decisions. The RS-232 communication standard is an example of a physical layer interface.

The data link layer provides an error-free transmission line to higher levels. Typically, it will divide input data into data frames. Each frame is transmitted independently. Acknowledgement frames from the destination provide confirmation of the correct arrival of a frame. Design issues at this layer include flow control, that is, making sure the sender does not overrun the input capacity of the receiver, and handling damaged, lost, and duplicate frames.

The network layer primarily controls the routing of information within the communication subnet. There are often multiple intermediate machines between the source and destination machines. Several possible data paths could be chosen to reach a given destination node. The network layer provides a level of control over the routing choices to the client machines. A node may not have complete control over routing choices. The communication subnet often makes many of the routing decisions.

The transport layer is a true host-to-host protocol. It is also called "source-to-destination" or "end-to-end." It is this layer which determines the type of communication subnet service available to upper layers. The two primary types of service are datagram and virtual circuit service. A virtual circuit is an error-free channel that delivers messages in the order in which they were sent. Datagram service delivers isolated messages with no guarantee of any particular arrival order. Broadcast service is sometimes provided as a primitive transport layer function, either by setting up multiple virtual circuits (requiring much overhead), routing a datagram to multiple destinations, or routing all messages to all destinations and letting the destinations choose which are appropriate. The transport layer also must establish and delete connections across the network. This includes some means of network-wide naming.

The session layer is the user's interface to the network. It includes authentication of network access, possibly recovery for transport layer problems, and for grouping of messages into atomic transactions. The operation of setting up a network communication between processes is often called binding.

The presentation layer performs services which are useful to a wide-variety of users, and therefore available in a common operating-system library. Text compression, conversion, and encryption are typical services. Virtual terminal and file transfer protocols are other typical presentation layer functions.

The application layer is determined by individual users. This includes any process to process agreements with respect to the allowable messages which can be communicated. It also includes large application packages such as database management systems with distributed capabilities, or industry specific protocols such as for banking or airline reservation systems.

7.2.2 Portions of OSI Model Applicable to CAIS

The following sections describe portions of the OSI Model that are applicable to CAIS.

7.2.2.1 Transport

Both virtual and datagram service should be provided by CAIS. Some form of broadcast service would be highly desirable as well. Virtual circuits have the potential of simplifying interfaces because they assume a reliable underlayer. The use a virtual circuits assumes some sort of circuit set-up operation (perhaps an OPEN), and explicit circuit termination (CLOSE). This is conceptually consistent with normal IO services. Datagram service does not require such an explicit setup and termination. It assumes each message is an isolated datum. Circuit set-up and termination can be expensive operations. Datagram service is considered more appropriate to transaction-oriented communication, while virtual circuits provide convenient bulk data transfer. Virtual circuits can be built on top of datagram service, but it makes no sense to attempt the converse. Virtual circuit and datagram service each have strong advocates. A good summary of the strengths of each approach is in section 6.6.3 of "Distributed Systems," B.W. Lampson, ed., Springer-Verlag, 1981 [11].

ISO has developed a proposal for the OSI transport layer interfaces. Their proposal should be carefully reviewed for applicability to CAIS, and should be followed unless it causes unresolvable problems in other areas of CAIS.

7.2.2.2 Session

The session layer binding involves not only physical communication setup and termination, but integrates access and security into the control process. ISO has also developed an OSI session layer protocol which should be examined for use by CAIS. Of particular interest will be the ability of the OSI protocol to accomodate CAIS access control and security rules. Since these are not fully defined, this becomes a somewhat recursive design problem.

7.2.2.3 Presentation

CAIS services for importing and exporting entities belong in the presentation layer of the model. Virtual Terminal support is at this level as well. CAIS already provides support for several classes of virtual terminals, with new classes currently proposed. File Transfer protocols would ordinarily be placed at this level as well. CAIS file transfers make no sense without associated type information. They are handled by the more powerful importing/exporting services. There is no OSI standard for presentation layer interfaces yet. There are several industry de facto standards for portions of the presentation layer. These would be applicable for communication with non-CAIS systems (if such communication is allowed). It is anticipated that such communication will be beyond the scope of the CAIS.

7.2.2.4 Application

This layer is considered to be beyond CAIS scope at this time. Some ISO standards have been developed for this level, such as X.400 E-mail. It would be interesting to examine the naming conventions and access controls for compatibility with CAIS. As more services are developed at this level it may become prudent to adopt them as CAIS standards.

7.3 CAIS BOUNDARIES

There is nothing inherent in the computational models of distribution, file system design choices, or operating system views of a distributed system which allow us to determine merely by examining physical characteristics if a given system is one instance of CAIS, or several instances. Some definition of what constitutes a CAIS must be agreed upon. This definition is enforced by software. Once a CAIS definition is available, it can then be determined what kinds of distribution activities are within a single CAIS instance (intra-CAIS) and which are among multiple CAIS instances (inter-CAIS).

7.3.1 Boundary Descriptions

A physical CAIS processor is defined to be a machine which can execute a CAIS process. A logical CAIS processor consists of one or more physical CAIS processors. A physical processor may or may not have storage devices attached to it, but is assumed to have access to a memory area. A CAIS database is a single rooted

ERA model database (or Semantic database if the design continues along those lines).

A single instance of CAIS must have:

a. One logical processor

b. One serial number generator for unique identifiers

c. One CAIS database

It is attractive to think of a type called CAIS. Each instance of the CAIS type would be the defining boundaries of an instance of CAIS. A rough template of the type CAIS follows:

a. At least one logical processor made of one or more physical processors. Each physical processor has a set of allowable operations (perhaps processes it can execute), and a set of IO devices (possibly null). Topological distribution of processors may be described.

b. One logical database, A single CAIS root node. This implies a well-defined set of entities, triggers, data descriptions, user/authorization data, and one schema. (Physical distribution of data is possible among the set of physical processors in this CAIS via relationships from structural nodes and/or classes to device nodes.)

c. One unique serial number generator

The sketches in Figure 7-8 show several representative CAIS boundaries. Sketch four shows especially well the futility of determining the boundaries of a CAIS by examining of physical characteristics alone. It also demonstrates a challenging problem of creating unique serial number generators for CAIS's which have no distinguishing physical characteristics.

7.4 Discussion of Specific Distribution Issues

The following sections describe specific distribution issues.

7.4.1 Remote Execution

CAIS will define remote execution as execution on another CAIS. This cannot be done without an import/export operation.

(An operation which may be trivial among CAIS's which are strongly related.) Probably, within a single CAIS some processors and portions of the database could be distributed such that network transport was required to use these resources. Although they would be physically remote, they would be logically part of a single CAIS and "remote" execution would be possible. Such "remote" execution may or may not use the same inter-CAIS communication services suggested by the OSI model. Intra-CAIS "remote" execution could be completely hidden from a requesting tool.

7.4.2 Remote Compiling and Run-time Systems

See separate issue report on Multiple Compiling Systems.

7.4.3 Using Differing Data Representations

This issue has been deferred.

7.4.4 Tools in a Distributed Environment

A distributed environment can refer to both intra-CAIS and inter-CAIS distribution. Inter-CAIS distribution requires both virtual circuit and datagram service with appropriate error recovery, optional visibility into lower layers of the communication protocols, and presentation, session, and application packages.

Intra-CAIS distribution could be completely hidden from tools. If the internal distribution is visible to tools, intra-CAIS services should resemble the inter-CAIS services (but would likely have different semantics). Implementations could support a wide-variety of local communication protocols, but if these are not hidden from tools, such tools may not be portable among other CAIS instances.

7.4.5 Tools Operating with Foreign Components

Can tools operate with files from foreign CPU's, from foreign compiling systems, with foreign run-time systems?

If foreign means "not within this CAIS instance," an import/export operation is required before foreign objects can be manipulated. This does not preclude processors of many types and compiling systems and run-time systems of many types resident within a single CAIS. The interoperability of tools within a single CAIS would be determined by the typing imposed within a particular CAIS. Type conversion routines may be locally available for using objects which are logically related, but produced by different tools. One example is Ada object files from different compilers, which could possibly be type converted for interoperability.

7.4.6 Transmitting Object Code Over the Network

Only if an import/export operation is performed. Again, the definition of remote is "outside of a single CAIS." Physically distributed execution over networks is possible within a single CAIS without importing or exporting.

7.4.7 Transported Data Reconstructing Entire Remote Data Structures

It would not be possible to reconstruct an entire remote data struture from transported data, assuming remote data structure means outside of this CAIS. Again, physically remote data structures within a single CAIS could be reconstructed by simple copy utilities, or if integrity of simultaneous remote updates is a concern with a particular data structure, replication services can be supported within a CAIS instance.

7.4.8 Current Standardization Feasibility

The definition of a CAIS, Transporting, and session services could be standardized, as could some presentation layer services such as to import/export. Other presentation services, such as encryption, seem beyond CAIS. Some application services are already defined by CAIS, such as Virtual terminal protocols, virtual mag tape, sequential IO, and direct IO.

7.4.9 Remote Tool Libraries

We certainly need libraries for importing and exporting tools. Two basic approaches are appropriate, both of which should be encouraged. One import/export service should reduce all type definitions and objects to some CAIS common external form. It should be a required service set on every CAIS. Another class of "short cut" import/export services should be available for systems with which a CAIS is expected to have either expensive or frequent operations but is not a member of the same CAIS instance. Such "short-cut" services may be CAIS dependent, but their syntactic interfaces should be identical to the standard form.

If "remote tool libraries" is interpreted as a network server of tools, then the utility of that approach is dependent on individual CAIS topologies, processing capabilities, and the difficulty of porting the tools.

7.4.10 Remote Relationships, Lifetime, and File Backups

There are no remote relationships except import and export which are both transient. They exist for the lifetime of the import or export operation and are then deleted. This avoids any problems with backing up databases with remote CAIS connections. Once again, "remote" refers to relationships outside this CAIS; physical distribution within a CAIS is still possible, and the services required to support such distribution can be substantial.

7.4.11 Services Needed to Make Distribution Workable

OSI services would be necessary, as mentioned above. A remote login would also be required. Notice that a remote login is not the same as remote execution, or file access, in that during a remote login session, the user is actually part of another CAIS. He may simultaneously be associated with more than one CAIS as a time, but operations on the remote CAIS are done as a member of that CAIS, not as a "remote" operation.

7.4.12 Services Needed by Tools

The transport layer and above should be accessible to all tools. Only special control tools probably need visibility into lower level network control such as routing decisions, control flow algorithms, error correction codes, and the like.

7.4.13 Scenarios Likely to be Practical in the Long Term

The User-Server Model seems to be gaining in popularity, and is expected to dominate local processing networks due largely to the performance-cost ratio of workstations. This computational distribution tends to favor a datagram-based transport service, for quick transactions between processors. However, long haul virtual circuit service is also growing rapidly. This means CAIS must do a good job of supporting both. More distribution is likely over the next 5 to 10 years. This favors a CAIS distributed operating system as opposed to a network operating system. The later would present a multitude of new translations to resolve if the "encapsulating" approach was used, or an intractable number of unique system interfaces for the user to master.

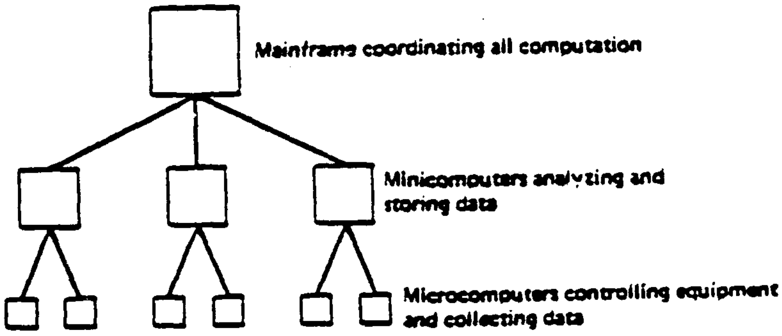

Mainframe coordinating all computation
Minicomputers analyzing and storing data
Microcomputers controlling equipment and collecting data

Figure 7-1. A Hierarchical Network.

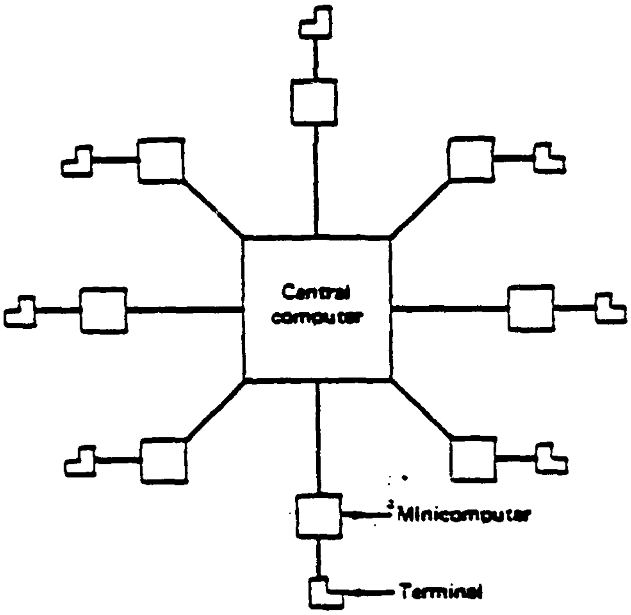


Figure 7-2. The CPU Cache Model.

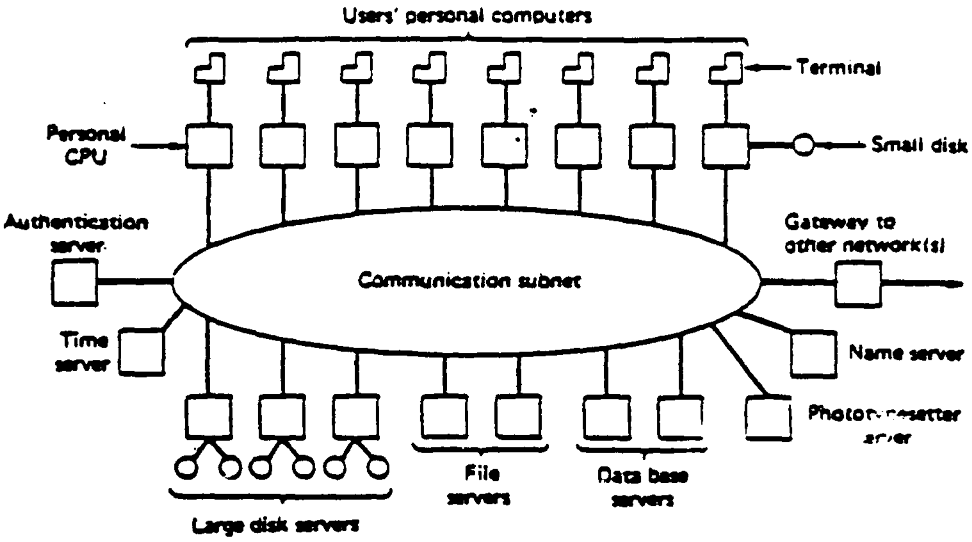


Figure 7-3. The User-server Model.

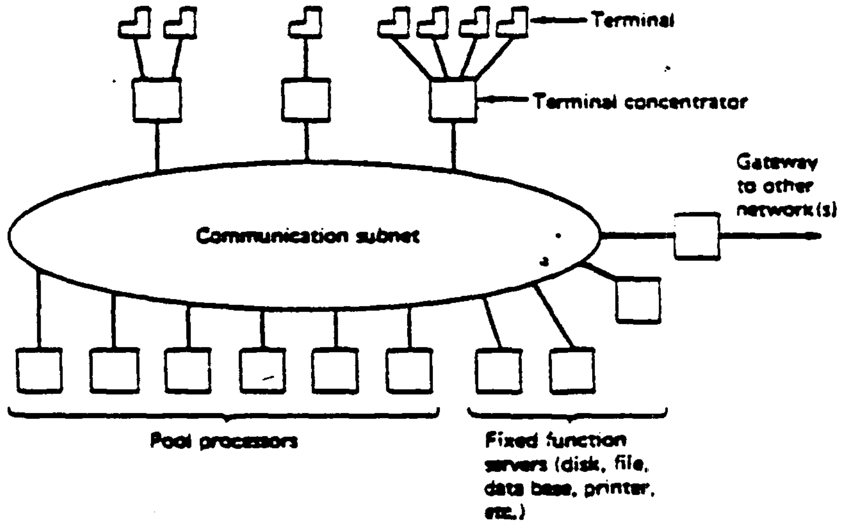

Terminal
Terminal concentrator
Gateway to other network(s)
Communication subnet
Pool processors
Fixed function servers (disk, file, data base, printer, etc.)

Figure 7-4. The Pool Processor Model.

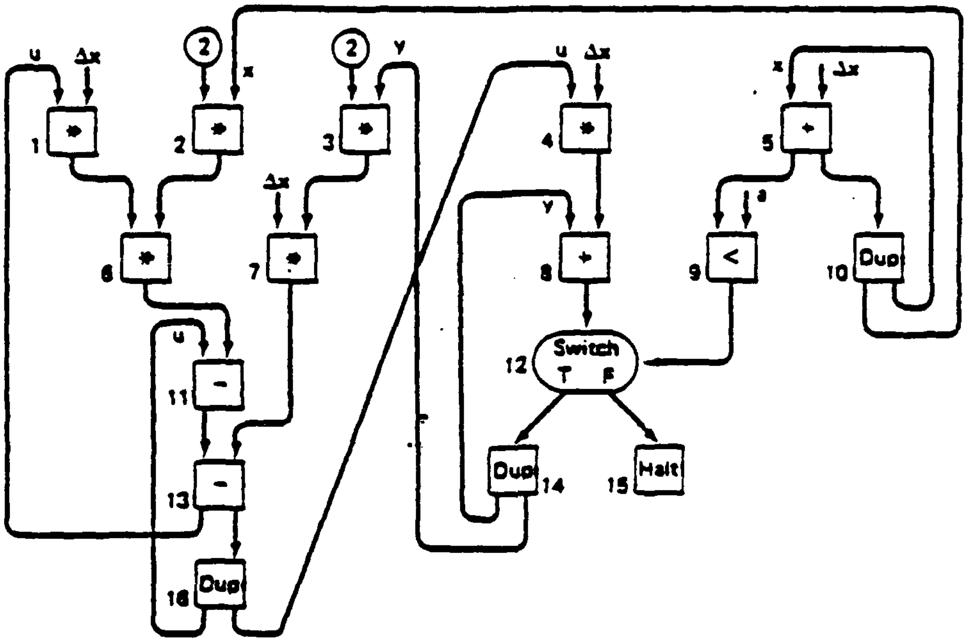
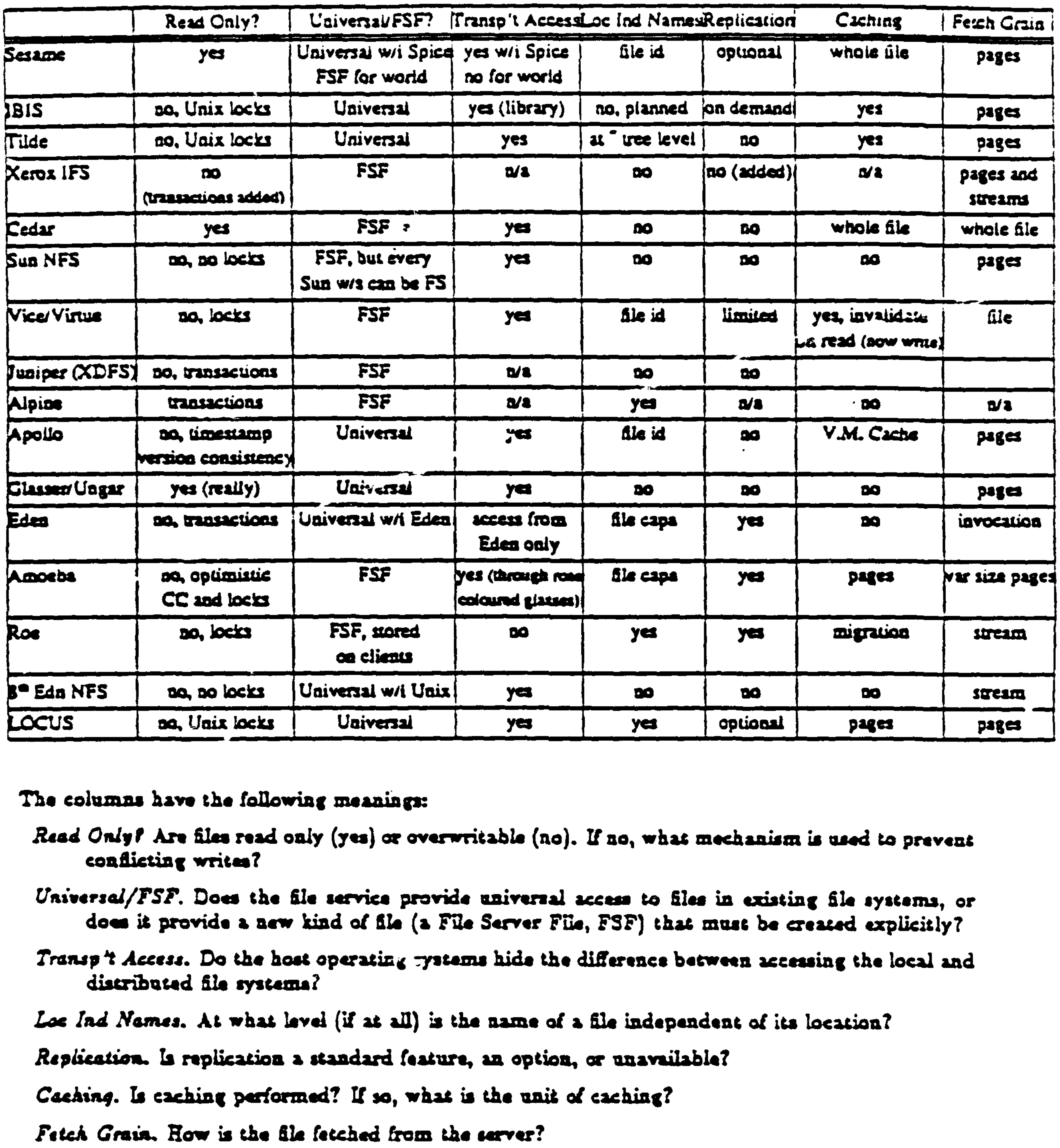


Figure 7-5. A Data Flow Graph.



	Read Only?	Universal/FSF?	Transp't Access	Loc Ind Names	Replication	Caching	Fetch Grain
Sesame	yes	Universal w/i Spice FSF for world	yes w/i Spice no for world	file id	optional	whole file	pages
IBIS	no, Unix locks	Universal	yes (library)	no, planned	on demand	yes	pages
Tilde	no, Unix locks	Universal	yes	at ˜ tree level	no	yes	pages
Xerox IFS	no (transactions added)	FSF	n/a	no	no (added)	n/a	pages and streams
Cedar	yes	FSF	yes	no	no	whole file	whole file
Sun NFS	no, no locks	FSF, but every Sun w/s can be FS	yes	no	no	no	pages
Vice/Virtue	no, locks	FSF	yes	file id	limited	yes, invalidate on read (now write)	file
Juniper (XDFS)	no, transactions	FSF	n/a	no	no		
Alpine	transactions	FSF	n/a	yes	n/a	no	n/a
Apollo	no, timestamp version consistency	Universal	yes	file id	no	V.M. Cache	pages
Glasser/Ungar	yes (really)	Universal	yes	no	no	no	pages
Eden	no, transactions	Universal w/i Eden	access from Eden only	file caps	yes	no	invocation
Amoeba	no, optimistic CC and locks	FSF	yes (through rose coloured glasses)	file caps	yes	pages	var size pages
Roe	no, locks	FSF, stored on clients	no	yes	yes	migration	stream
8ᵗʰ Edn NFS	no, no locks	Universal w/i Unix	yes	no	no	no	stream
LOCUS	no, Unix locks	Universal	yes	yes	optional	pages	pages

The columns have the following meanings:

Read Only? Are files read only (yes) or overwritable (no). If no, what mechanism is used to prevent conflicting writes?

Universal/FSF. Does the file service provide universal access to files in existing file systems, or does it provide a new kind of file (a File Server File, FSF) that must be created explicitly?

Transp't Access. Do the host operating systems hide the difference between accessing the local and distributed file systems?

Loc Ind Names. At what level (if at all) is the name of a file independent of its location?

Replication. Is replication a standard feature, an option, or unavailable?

Caching. Is caching performed? If so, what is the unit of caching?

Fetch Grain. How is the file fetched from the server?

Figure 7-6. Comparison of File Systems.

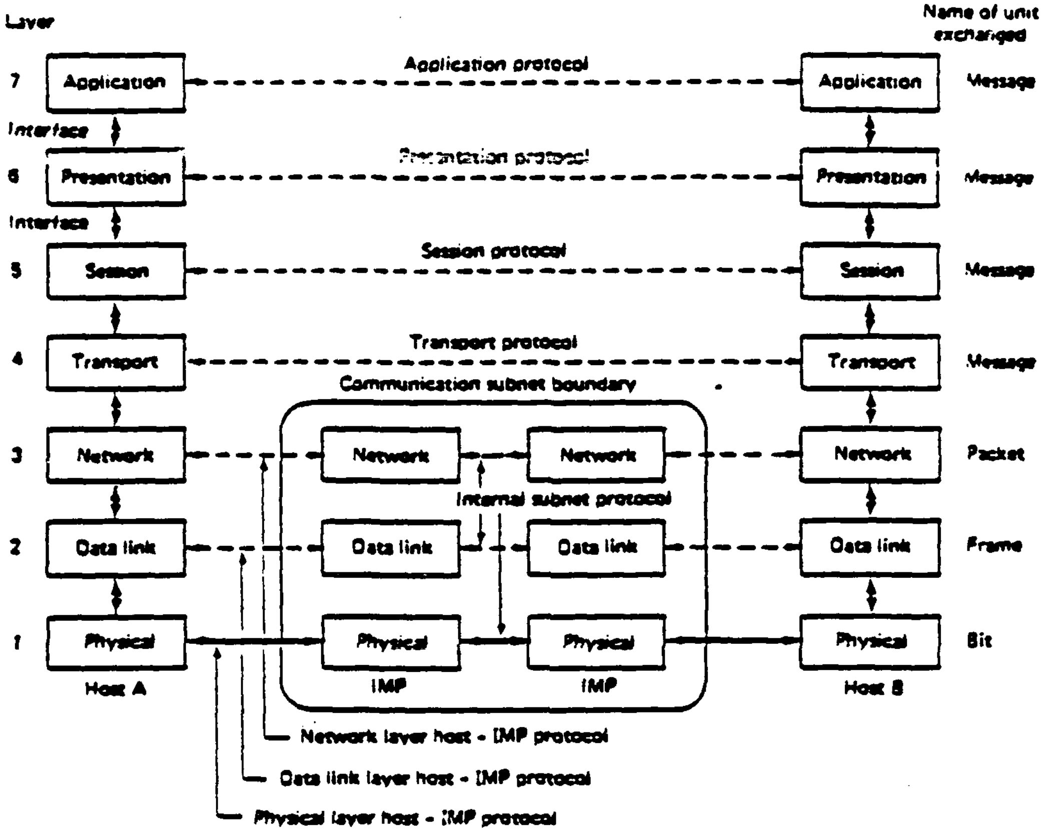
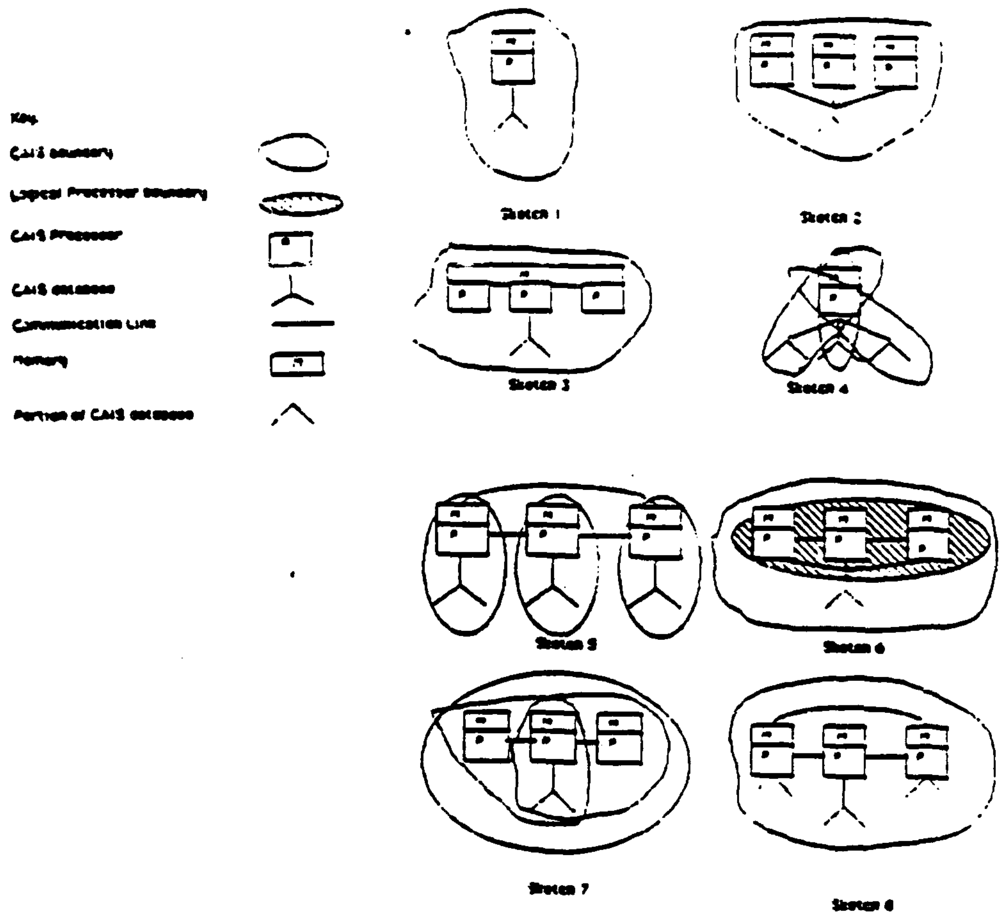


Figure 7-7. The ISO-OSI Model.



Sketch 1 - A simple CAIS

Sketch 2 - A CAIS with multiple processors

Sketch 3 - Another CAIS with multiple processors
Note: Memory sharing does not change CAIS boundary
from sketch 2

Sketch 4 - Three CAISes with a common processor

Sketch 5 - Three CAISes which are linked together

Sketch 6 - One CAIS with individual processors hidden from users
Compare to sketch 2

Sketch 7 - Four CAISes. Note that each must have a database

Sketch 8 - One CAIS. Note that database must be whole. Compare to
sketch 7

Figure 7-8. Several Representative CAIS Boundaries.

ISSUE SUMMARY

ISSUE REPORT REVISION HISTORY

ISSUE 8.0 -- Transaction Issues

REV		SECTION	
NUMBER*	DATE	REVISED*	DESCRIPTION OF REVISION
0	6/30		First Draft

* Since sections may be added and deleted, section numbers of changed and deleted sections apply to the previous revision only. Numbers of added sections apply to the latest revision.

MAJOR ISSUE: How are transactions to be supported in the CAIS?

SECTION/REV REFERENCE	STATUS	DESCRIPTION OF ISSUE
8.0	Transaction Issues	
8.0/0	OPEN	What are the bounds of a transaction? Can one transaction span: 1. Ada tasks? 2. processes? 3. sessions?
8.0/0	OPEN	What are the implications of each to the implementor?
8.0/0	OPEN	Can transactions be combined with triggers?
8.0/0	OPEN	In a distributed CAIS, can transactions span nodes?

CHAPTER 8

TRANSACTION ISSUES

This section introduces the CAIS 2 requirements associated with transactions and provides a short definition of transactions.

NOTE

> This chapter is presented in a very preliminary form.

8.1 DESCRIPTION OF A TRANSACTION

A transaction is a sequence of primitive actions packaged together in such a way that either all of the primitive actions take place or none do. A transaction is also called a compound atomic action. It is a unit of recovery and resource allocation that is scheduled independently of other transactions. The following are components of a transactional system.

a. Transaction scheduler (TS) - the transaction scheduler component is the active part of a transactional system. It schedules creation and deletion of transactions, allocates resources for new work, maintains the dispatcher list, and dispatches.

b. Concurrency Control - the concurrency control component ensures that data consistency is maintained between transactions. That is, the concurrency control component must ensure that if two transactions are executing concurrently, then their input and output sets do not interfere with each other.

c. Data Management - the data management component organizes data into more complex structures. In transactional systems, the data manager is considered to be a separate component because all of the other components must interact extensively with the data manager. These interactions involve the concurrency control component, the logging component, and the recovery component.

d. Logging - the logging component maintains a history of what happens to the objects managed by the data manager. The recovery component uses this history.

e. Recovery - the recovery component ensures consistency between the data management component and the logging component. It is invoked when a transaction is aborted, such as during system shutdown and system restart.

8.2 CAIS TRANSACTIONAL REQUIREMENTS

The following requirements are described in the RAC.

a. Transaction Mechanism - The CAIS shall support a transaction mechanism. The effect of running transactions concurrently shall be as if the concurrent transaction were run serially.

b. Transaction Control - The CAIS shall support facilities to start, end and abort transactions. When a transaction aborts, all effects of the designated sequence of operations shall be as if the sequence were never started.

c. System Failure - System failure while a transaction is in progress shall cause the effects of the designated sequence of operations to be as if the sequence were never started.

8.3 TRANSACTION SCHEDULING

Transaction scheduling comprises the startup, execution, and rundown of transactions. The following sections describe these three items.

8.3.1 Transaction Objects

Transaction scheduling manipulates three different kinds of objects: the descriptor, the process, and the instance.

8.3.1.1 Descriptor

The transaction descriptor describes how to build an instance of a transaction.

The descriptor typically has the following information:

a. Scheduling information

b. Recovery information

c. Concurrency control parameters

d. Data management information

e. Process information (described below)

8.3.1.2 Process Information

Processes are responsible for executing transactions. A process is bound to a sequence of executable code and to other resources. For the purpose of the transaction scheduler, a process is a unit of scheduling and resource allocation.

8.3.1.3 Transaction Instance

A transaction instance is the unit of concurrency control and recovery.

8.3.1.4 Intermingling of Instances

Instances intermingle when more than one transaction instance is active at a time. One process may have many transaction instances (intermingling), and of course, processes intermingle.

8.3.2 Transaction Lifetime

A transaction's lifetime begins with a start transaction command and ends with a commit transaction or abort transaction command. The following options are available:

a. A process may establish the lifetime by issuing the start and commit or abort commands.

b. The scheduler may issue start and commit commands on behalf of the process, bracketing the execution of this process.

8.3.3 Composition of a Transaction

The composition of a transaction is as follows:

a. Processes - a transaction can be composed of one or many processes.

b. Client communications -

   1. One message in and one message out

   2. Many messages in and many messages out

   3. None

c. Cohort communications - none or many messages.

d. start/commit pairs - one or many

8.3.4 Scheduler Activities

8.3.4.1 System Startup

The TS is involved in system startup for the following reasons.

a. Predeclared transaction must be started up

b. Other - TBD

8.3.4.2 System Shutdown

a. Stop scheduling used transactions.

b. Invoke special shutdown transactions.

c. Notify recovery component that all is shut down

8.3.4.3 Checkpointing

Record current scheduling status

8.3.4.4 Listening for New Work

The TS listens for new work. It takes the following into consideration when processing a request:

a. Overloaded System - if the system is overloaded, the request may be delayed or rejected.

b. Limit on Type of Transaction - the scheduler may place a limit on the types of transaction that may be active at any one time.

c. Thresholding - the TS may hold new work requests until a certain number of requests are enqueued.

d. Unavailable resources - if a transaction requests unavailable resources, it may be delayed or rejected.

e. Special times - certain new work requests may be run at special times.

8.3.4.5 Allocating Resources for New Work

The TS allocates resources for new work by interfacing with the other components of the system as follows:

a. Process allocation - if the request requires process allocation, this is the point at which the allocation occurs. Depending on what constitutes a process, the TS creates one of the following:

1. An Ada task,
2. An Ada program,
3. A session, or
4. An ASE.

b. Data Management

1. cursors are established if necessary
2. files are opened.

c. data communications - queues are established as necessary

d. recovery component - is notified of a new transaction starting up.

e. concurrency component - is notified of a new transaction starting up.

f. logging component - TBD

g. others

8.3.4.6 Running Down Transactions

The TS listens for transaction rundown commands. Rundown commands may be issued by:

a. the process or its cohorts. This command may be a commit or an abort.

b. the concurrency control component. This is usually due to a concurrency error, resulting in an abort.

c. the system scheduler. This is due to an error which is forcing the ASE to go away.

8.3.4.7 Deallocating Resources for Commit or Abort Processing

This is the reverse of the allocation and its actions depend on whether the transaction is committing or aborting.

8.3.4.8 Maintenance of Dispatcher List

The TS provides facilities to maintain the dispatcher list. This list determines which transactions are blocked and which may run Each component of the transaction system decides when a transaction may run and when it is blocked and may not run. The following components use the dispatcher list:

a. Concurrency control - blocks a transaction when an object is not available, that is, it is being used by another transaction.

b. Data management - blocks a transaction when the requested data are not available in memory.

c. recovery - TBD

d. logging component - the recording of logging information is not complete.

8.3.4.9 Dispatching and Interfacing with System Dispatcher

The question here: is the transaction scheduler the same as the system scheduler? The cases are:

8.3.4.9.1 Transaction Scheduler and System Scheduler

What are the questions when they are the same?

When they are not the same, the questions are

1. Can the transaction be blocked and the ASE be blocked?
2. Can the transaction be blocked while the ASE is runnable?
3. Can the transaction be runnable while the ASE is blocked?

4. Can the transaction be runnable while the ASE is runnable?

8.3.4.10 Other Scheduling Issues

a. primed transactions - transactions that are ready and waiting for work. Corresponds to process reuse in the KAPSE.

b. batched messages - a process looks for more work before terminating. This implies that a process may issue numerous start/commits.

8.4 CONCURRENCY CONTROL

a. degrees of consistency control

b. kinds of data dependencies

8.4.1 Pre-execution Analysis

TBD

8.4.2 Post-execution Analysis

TBD

8.4.3 Locking

TBD

8.5 DATA MANAGEMENT

TBD

8.6 LOGGING

TBD

8.7 RECOVERY

TBD

APPENDIX A

APPENDIX A

AN APPROACH TO STRONG TYPING OF ENTITIES IN CAIS 2

This material was supplied by Robert G. Munck of The Mitre Corporation (617-271-3671).

A.1 OVERVIEW

The following is a proposal for the CAIS 2 Entity Management System that seems to meet the RAC requirements in a simple, straightforward, and Ada-like way.

The proposal is to use an Ada package specification as the definition of a CAIS type and the package body as the implementation of that type. This will make entity typing in CAIS 2 a real, usable, and easy-to-understand concept and should, in most systems, be implementable with very good performance.

A.2 THE PROPOSAL

It is proposed that the type of an entity be defined by an associated Ada package specification and body. The specification would describe the subprogram and entry calls, data types, and exceptions that user code must use to access all entities of that type. The package body would provide access to the entity. The package is called a "type handler."

The compiled type handler specification is used when a user tool is compiled; the compiled and linkable or loadable body is used either at tool link time or when the entity is opened during tool execution.

A.3 TYPING

An individual type handler specification would determine what attributes and relationships an entity of that type can have, by giving the information needed by tool code to access them. A simple type similar to an ordinary file might have OPEN, READ, WRITE, and CLOSE entries to access an uninterpreted attribute named "Contents", entries named "Date" and "Size" to retrieve other attributes, and entries named "Owner", "Parent", and "PreviousVersion" to retrieve relationships. It might define exceptions named END_FILE and HARDWARE_ERROR.

Because a tool may need to access several entities of the same type simultaneously, the concept of an "entity handle" is still necessary. It would include as private data whatever information the type handler needs concerning the current state and location of the entity (disk address, buffer, current position, etc.).

A.3.1 CAIS-Defined Types

CAIS 2 would specify a set of primitive or "built-in" type handlers, attribute types, and relationship types. For example, it would probably describe a "sequential disk" handler that provides a sequentially-accessed list of storage units stored on disk. Other pre-defined entity types would include a printer, terminal, tape, and other kinds of disk storage. Likewise, there would be pre-defined attributes such as Date and relationships such as Owner.

There would also be a number of CAIS-defined types for supported I/O devices. The type handler would be the equivalent of a device driver.

A.3.2 User-Defined Types

A new type handler is implemented by using existing, lower-level handlers, either simply by renaming entries from the handler used or by writing code to do the necessary manipulations. For example, a QUEUED_PRINTER type might use entities of type SEQUENTIAL_DISK and PRINTER to support printer queuing.

A type handler that re-exports (renames) all of the specification of another handler (in addition to implementing new items) is said to be "upward compatible" with it. When the new tool specification is compiled, the tool-writer would stipulate this fact, the compiler would check it by looking at the specification of the old tool, and record it in the stored specification of the new tool. At the time that the linkage between a tool and a handler completes this record would be used to allow the tool to use the higher-level handler even if it was compiled to use the lower. This satisfies the "type lattice" requirements of the RAC.

A.4 LINKING

In one possible implementation, a tool would be compiled and linked using the specification only, without including the body code in the link module; calls from the tool to the type handler would be compiled to trap to the OS. When the executing tool does an OPEN for the entity, the body is loaded into memory and the traps are replaced with direct calls between the tool and handler body. (Note that this is essentially how MULTICS, TSS, and other implementations of dynamic linking work.) Type handlers would be written to be reentrant (sharable); the most frequently-used handlers would be kept in main memory to minimize OPEN time.

Where the Ada system or underlying OS do not permit this kind of dynamic linking, the tool may need to be combined with all of its type handlers at link time and stored that way. Note, however, that this implementation choice does not impact the functionality of the system as seen by users, only the performance.

A.4.1 Type Structure

Every entity in the data base would have a relationship to its type handler, which are also entities themselves. All varieties of system-wide type definition (type handlers in system libraries) and user-private type definition (type handlers in private libraries) would therefore be possible. Each type handler would have relationships to all other type handlers with which it is upward compatible, including earlier revisions of itself; each tool would have relationships to the type handlers that it uses. In order for a tool to access an entity, it is necessary that the tool's relationship to the type handler and the entity's relationship to its type handler be identical, or that an identical relationship can be found by following upward-compatible relationships from the entity's type handler. The basic structure would be:

```
+-------+
|       |  UsesType
| tool  +-->--------+
|       |           |
+-------+       +---+---+
                | type  |
                |handler|
                |   x   |
+-------+       +---+---+
|entity |           |
|  of   +-->--------+
|type x |  IsType
+-------+
```

The UsesType relationship is created when the tool is installed in the system; the IsType relationship is created when the entity is. When the tool attempts to access the entity, UsesType and IsType are compared to see if they point to the same thing. If so, the body of type handler x is loaded into memory and the links from tool to it are resolved. A more complex case:

```
+-------+
|       |  UsesType
| tool  +-->--------+
|       |           |
+-------+       +---+---+
 IsCompatible   | type  |
    +------->---+handler|
    |           |x rev 1|
+---+---+       +-------+
| type  |  IsCompatible
|handler+--<--------+
|x rev 2|           |
+-------+       +---+---+
                | type  |
                |handler|
                |   y   |
+-------+       +---+---+
|entity |           |
|  of   +-->--------+
|type y |  IsType
+-------+
```

The tool was installed using type handler x rev 1; type handler x was later changed by adding facilities, leaving those in rev 1 unchanged, to make rev 2. A new type handler named y was then created from x that also only added facilities. When the tool tries to access the entity, the tree of IsCompatible relationships would be searched for the object pointed to by the tool's UsesType relationship.

A.4.2 Entity Transportability And Backup

Entities stored on disk could have a "bottom-most" type handler that stores all attributes as a single list-valued attribute containing lists of strings and all relationships as a single list-valued relationship. Backup and Restore tools would use this handler. Note that it is in some sense "below" the level of the CAIS, in that CAIS-specified types could also be implemented with handlers that use it.

Another possibility is that all type handlers could have a ReadEntity entry point that returns the complete "contents" (all attributes and relationships) of the entity in ASCII character stream form. There would also be a WriteEntity entry point that accepts such a character stream and creates or recreates the entity. The internal structuring or ordering of the character stream would be whatever the writer of the type handler finds usable. The stream could be stored for backup or transmitted to another instance of the type handler on another machine.

A.5 NOTES

1. The proposed approach seems (to me) to be very much in the "spirit" of Ada. It is highly dependent on Ada and exploits many of its strengths.

2. This proposal could support extremely flexible, or <u>undisciplined</u>, environments; many of the integrity features called for in the RAC would be implemented (or omitted) in the type handlers. This does <u>not</u> mean that a particular installation cannot have very good integrity; only that the CAIS 2 specification does not set in stone our current ideas of what integrity means. A particular installation would enforce their own concept of integrity by setting programming standards for what a type handler must, can, and cannot do. For example, the following are possible rules:

   a. All attributes containing date and time values must use the DateTime type definition from the SystemDataTypes library.

b. All type handlers must have ReadEntity and WriteEntity entry points.

c. All type handlers must create relationships to and from the User entity of its owner when creating a new entity.

d. The Create entry of type handler User must verify that it is being run under a User with attribute Administrator=TRUE. Setting attribute Administrator requires the same check.

3. These checks can be made at the time that a new type handler is installed in the system type library, either mechanically or by inspection. Type handlers are extensions of the CAIS implementation, and therefore must be written and installed with appropriate care.

4. Type handlers will tend to accumulate layers as new types are implemented on top of old, introducing processing overhead. However, a popular type that is degraded by too many layers can be speeded up by the writing of a new body that uses the primitive or built-in type handlers directly, bypassing the accumulated layers.

5. Compatibility with CAIS 1 can be maintained to whatever degree is desired by making the CAIS-specified types "look like" the CAIS 1 calls.

6. There is a strong interplay between this proposal and the requirements for process control. In fact, most of the pieces of CAIS 1 are subsumed by this single mechanism. Most other RAC requirements (process model, security) should be easy to integrate with this approach.

7. This proposal makes it necessary that a single compiler be used for compilation of a CAIS implementation and all tools running on it (though other compilers might be usable with appropriate representation specifications). This is probably not an important limitation. A mechanism should be devised whereby a vendor can supply tools to a buyer without needing to supply source code.

APPENDIX B

REFERENCES

[1] KAPSE Interface Team (KIT) and KIT-Industry-Academia (KITIA) for the Ada Joint Program Office; DoD Requirements and Design Criteria for the Common APSE Interface Set (CAIS); 13 September 1985.

[2] P. P. Chen, "The entity-relationship model: Towards a unified view of data", ACM Transactions on Database Systems, vol. 1, no. 1, 1976.

[3] Military Standard Common APSE Interface Set (CAIS); Proposed MIL-STD-CAIS, 31 January 1985.

[4] BULL, GE, etc.; PCTE - A Basis for a Portable Common Tool Environment, Functional Specification, 3rd ed., vol. 1, 1985.

[5] American National Standards Institute; ANSI/X3H4 Annexes 1A-1E, November 1984.

[6] Proceedings of the Department of Defense Computer Security Center Invitational Workshop on Network Security, New Orleans, LA, 19-22 March.

[7] RAC/CAIS 1 Comparison and Analysis 1138-1-2, SofTech, Inc, NOSC Contract No. N66001-86-0101, 25 April 1986.

[8] Department of Defense Requirements and Design Criteria for the Common APSE Interface Set (CAIS), KIT/KITIA, 13 September 1985 (Draft).

[9] "Computer Networks," Andrew S. Tanenbaum, Prentice Hall, 1981.

[10] The Report on the ACM SIGOPSWorkshop on Accomodating Heterogeneity by David Notkin, et al, Technical Report 86-02-01, Dept. of Computer Science, Univ. of Washington, March 1986.

REFERENCES

[11] "Distributed Systems," B.W. Lampson, ed., Springer-Verlag, 1981.

DoD

Requirements

and

Design Criteria

for the Common APSE Interface Set (CAIS)

14 July 1986

This draft of this document is NOT APPROVED
for distribution outside KIT/KITIA and the AJPO.

Prepared by the
KAPSE Interface Team (KIT)
and the
KIT-Industry-Academia (KITIA)

for the
Ada* Joint Program Office (AJPO)
Washington, D.C.

*** Ada is a Registered Trademark of the U.S. Government,**
Ada Joint Program Office

Table of Contents

PREFACE

The KAPSE Interface Team (KIT), and its companion Industry-Academia team (KITIA), were formed by a Memorandum of Agreement (MOA) signed by the three services and the Undersecretary of Defense in January, 1982. Their purpose is to contribute to the achievement of Interoperability of environment databases (of applications software) and Transportability of software development tools ("I&T"). These are economic objectives identified at the outset of the DoD common language initiative in the mid-1970's. Progress toward fulfilling these objectives is now acknowledged to require a level of commonality among Ada Programming Support Environments (APSEs), in addition to the standard language Ada [Ada83]. The core of the KIT/KITIA strategy to fulfill I&T objectives is to define a standard set of APSE interfaces ("CAIS" for "Common APSE Interface Set"), which augment the Ada language with the functionality needed to implement tools, thus improving the ability to share tools and databases between conforming APSEs. Note that a number of these interfaces are at the Kernel APSE (KAPSE) level, while others address a higher level of functionality. This document establishes requirements and design objectives (called "criteria") on the definition of a CAIS.

This document refines some of the DoD "Stoneman" Requirements for Ada Programming Support Environments [Buxton80] and imposes them upon a CAIS specification. The DoD "Steelman" Requirements for High Order Computer Programming Languages [Fisher78] and the several sets of ANSI "OSCRL" requirements and design objectives for Operating System Command and Response Languages [OSCRL82] have also influenced this document.

1. INTRODUCTION

1.1 Scope. This document provides the Department of Defense's requirements and design criteria for the definition and specification of a Common APSE Interface Set (CAIS) for Ada Programming Support Environments (APSEs).

1.2 Terminology. Precise and consistent use of terms has been attempted throughout the document.

Potentially ambiguous terms used in the document are defined in the Glossary of KIT/KITIA Terminology [KK85]. Some definitions tailored to the context of this document are provided in the sections of the document where they are used.

Additionally, the following verbs and verb phrases are used throughout the document to indicate where and to what degree individual constraints apply. Any sentence not containing one of the following verbs or verb phrases is a definition, explanation or comment.

"SHALL" indicates a requirement on the definition of the CAIS; sometimes "shall" is followed by "provide" or "support," in which cases the following two definitions supersede this one.

"SHALL PROVIDE"
indicates a requirement for the CAIS to provide interface(s) with prescribed capabilities.

"SHALL SUPPORT"
indicates a requirement for the CAIS to provide interface(s) with prescribed capabilities or for CAIS definers to demonstrate that the capability can be constructed from CAIS interfaces.

"SHOULD" indicates a desired goal (design criterion) but one for which there is no objective test.

1.3 Relationship to Specifications & Implementations. This document specifies functional capabilities which are to be provided in the semantics of a CAIS specification and are therefore to be provided by conforming CAIS implementations. In general, the specifications of software fulfilling those capabilities (and decisions about including or not including CAIS interfaces for certain capabilities as suggested by the "shall support" definition in the previous section) are delegated to the CAIS definers. If a CAIS implementor determines that it is feasible, then the CAIS implementor may provide a particular specified CAIS facility by reusing other CAIS facilities, thereby achieving a "layered implementation" of the CAIS. Therefore, the realization of a specific CAIS implementation is the result of intentionally divided decision-making authority among 1) this requirements document, 2) CAIS definers, and 3) CAIS implementors.

1.4 Reference Documents.

MILITARY STANDARDS

[Ada83] Reference Manual for the Ada Language, ANSI/MIL-STD-1815A, January 1983.

OTHER GOVERNMENT DOCUMENTS

[Buxton80] "Stoneman" DoD Requirements for Ada Programming Support Environments, February 1980.

[Fisher78] "Steelman" DoD Requirements for High Order Computer Programming Languages, June 1978.

[KK85] Glossary of KIT/KITIA Terminology, draft 1985.

[TCSEC83] Trusted Computer System Evaluation Criteria, CSC-STD-001-83, DoD Computer Security Center, August 15, 1983.

NON-GOVERNMENT DOCUMENTS

[OSCRL82] Operating System Command and Response Languages, proposed ANSI standard drafts, 1982.

2. GENERAL DESIGN OBJECTIVES

The following definitions, used in this section, pertain to the remainder of the document also:

FACILITY a service obtained by calling one or a combination of multiple CAIS interfaces.

INTERFACE a specification of a CAIS subprogram callable by APSE tools.

2.1 Scope of the CAIS. The CAIS shall provide interfaces sufficient to support the use of APSEs for wide classes of projects throughout their lifecycles and to promote I&T of tools and data between APSEs.

2.2 Basic Services. The CAIS should provide simple-to-use mechanisms for achieving common, simple actions. Facilities which support needs of less frequently used tools should be given secondary consideration.

2.3 Implementability. The CAIS specification shall be machine independent and implementation independent. The CAIS shall be implementable on bare machines and on machines with any of a variety of operating systems. The CAIS shall contain only interfaces which provide facilities which have been demonstrated in existing commercial or military software systems. CAIS features should be chosen to have a simple and efficient implementation in many machines, to avoid execution costs for unneeded generality, and to ensure that unused portions of a CAIS implementation will not add to execution costs of a non-using tool. The measures of the efficiency criterion are, primarily, minimum turnaround time for CAIS basic services used by APSE tools and, secondarily, consumption of resources.

2.4 Modularity. Interfaces should be partitioned such that the partitions may be understood independently and there are no undocumented dependencies between partitions.

2.5 Extensibility. The design of the CAIS should facilitate development and use of extensions of the CAIS; i.e., CAIS interfaces should be reusable so that they can be combined to create new interfaces and facilities.

2.6 Technology Compatibility. The CAIS shall adopt existing standards where applicable. For example, recognized standards for device characteristics are provided by ANSI, ISO, IEEE, and DoD.

2.7 Uniformity. A small number of unifying conceptual models should underlie the CAIS. All CAIS features should uniformly address aspects such as status returns, exceptional conditions, parameter types, and options.

2.8 Security. The CAIS shall provide interfaces to allow tools to operate within a Trusted Computer System (TCS) that meets the Class B3 criteria as defined in [TCSEC83]. Specifically:

a. It shall be possible to implement the CAIS within a TCS.

b. When implemented within a TCS, the CAIS shall support the use of the security facilities provided by the Trusted Computing Base (TCB) to applications programs.

c. When not implemented within a TCS, the CAIS interfaces sensitive to security shall operate as a dedicated secure system (i.e., all data at a single security level, and all subjects cleared to at least that level).

2.9 Visible Distribution of CAIS Facilities

2.9A Reference. The CAIS shall provide a means for tools to refer to distinct physical resources (both computational and storage) that are used to implement specific CAIS facilities.

2.9B Reallocation. The CAIS shall provide a means to control (or influence) the manner in which the physical resources are associated with specific CAIS facilities.

3. GENERAL SYNTAX AND SEMANTICS

3.1 Syntax

3.1A General Syntax. The syntax of the CAIS shall be expressed as Ada package specifications. The syntax of the CAIS shall conform to the character set as defined by the Ada standard (section 2.1 of ANSI/MIL-STD-1815A [Ada83]).

3.1B Uniformity. The CAIS should employ uniform syntactic conventions and should not provide several notations for the same concept. CAIS syntax issues (including, at least, limits on name lengths, abbreviation styles, other naming conventions, relative ordering of input and output parameters, etc.) should be resolved in a uniform and integrated manner for the whole CAIS.

3.1C Name Selection. The CAIS should avoid coining new words (literals or identifiers) and should avoid using words in an unconventional sense. Ada identifiers (names) defined by the CAIS should be natural language words or industry accepted terms whenever possible. The CAIS should define Ada identifiers which are visually distinct and not easily confused (including, at least, that the CAIS should avoid defining two Ada identifiers that are only a 2-character transposition away from being identical). The CAIS should use the same name everywhere in the interface set, and not its possible synonyms, when the same meaning is intended.

3.1D Pragmatics. The CAIS should impose only those restrictive rules or constraints required to achieve I&T. CAIS implementors will be required to provide the complete specifications of all syntactic restrictions imposed by their CAIS implementations.

3.2 Semantics

3.2A General Semantics. The CAIS shall be completely and unambiguously defined. The specification of semantics should be both precise and understandable. The semantic specification of each CAIS interface shall include a precise statement of assumptions (including execution-time preconditions for calls), effects on global data and packages, and interactions with other interfaces.

3.2B Repeatability. Every time a CAIS interface is called under the same circumstances, it should return the same response.

3.2C Exceptions. The CAIS interfaces shall employ the mechanism of Ada exceptions to report exceptional situations that arise in the execution of CAIS facilities. The CAIS specification shall include exceptions (with visible declarations) for all situations that violate the preconditions specified for the CAIS interfaces. The CAIS specification shall include exceptions (with visible declarations) that cover all violations of implementation-defined restrictions.

3.2D Consistency. The description of CAIS semantics should use the same word or phrase everywhere, and not its possible synonyms, when the same meaning is intended.

3.2E Cohesiveness. Each CAIS interface should provide only one function.

3.2F Pragmatics. The CAIS specification shall enumerate all aspects of the meanings of CAIS interfaces and facilities which must be defined by CAIS implementors. CAIS implementors will be required to provide the complete specifications for these implementation-defined semantics.

4. ENTITY MANAGEMENT SUPPORT

Access controls and security rights will apply to all CAIS facilities required in this section.

The general requirements for the CAIS entity management support are the following.

a. There shall be a means for retaining data.

b. There shall be a way for retaining relationships among and properties of data.

c. There shall be a way of operating upon data, deleting data, and creating new data.

d. There shall be a means for defining certain operations and conditions as legal, for enforcing the definitions, and for accepting additional definitions of legality.

e. There shall be a means to describe data, and there shall be a means to operate upon such descriptions. Descriptions of the data shall be distinguished from the data described.

f. There shall be a way to develop new data descriptions by inheriting (some of) the properties of existing data descriptions.

g. The relationships and properties of data shall be separate from the existence of the data instances.

h. The descriptions of data and the instances of data shall be separate from the tools that operate upon them.

i. The data facilities shall be sufficient to support Ada program libraries.

This characterization (subsections 4.1 - 4.7) of Entity Management Support is based on the STONEMAN requirements for a database, using a model based on the entity-relationship concept. Although a CAIS design meeting these requirements is expected to demonstrate the characteristics and capabilities reflected here, it is not necessary that such a design directly employ this entity-relationship model.

The entity-relationship model, for which definitions and requirements follow in 4.1 - 4.7, fulfills these requirements, and any alternative dat model shall fulfill these requirements and shall also fulfill the equivalent of the requirements in 4.1 through 4.7.

4.1 Entities, Relationships, and Attributes The following definitions, used in this subsection, pertain to all the rest of section 4 also:

ENTITY a representation of a person, place, event or thing.

RELATIONSHIP an ordered connection or association among entities. A relationship among N entities (not necessarily distinct) is known as an "N-ary" relationship.

ATTRIBUTE an association of an entity or relationship with an elementary value.

ELEMENTARY VALUE
one of two kinds of representations of data: interpreted and uninterpreted.

INTERPRETED DATA
a data representation whose structure is controlled by CAIS facilities and may be used in the CAIS operations. Examples are creation date, revision count, and record size.

INTEGRITY preservation of conformance of the structure and contents of data tc rules established by a particular APSE as defined by the CAIS specification, implementation-defined CAIS values and parameters, APSE administrators, and users.

UNINTERPRETED DATA
a data representation whose structure is not controlled by CAIS facilities and whose structure is not used in the CAIS operations. Examples might be representations of files, such as requirements documents, program source code, and program object code.

4.1A Data. The CAIS shall provide facilities for representing data using entities, attributes or binary relationships. The CAIS may provide facilities for more general N-ary relationships, but it is not required to do so.

4.1B Elementary Values. The CAIS shall provide facilities for representing data as elementary values.

4.1C System Integrity. The CAIS facilities shall ensure the integrity of the CAIS-managed data. Some of these facilities are access control, concurrency control, database backup and restoration, and transactions.

4.2 Typing The following definition, used in this subsection, pertains to the remainder of section 4 also:

TYPING an organization of entities, relationships and attributes in which they are partitioned into sets, called entity types, relationship types and attribute types, according to designated type definitions.

4.2A Types. The facilities provided by the CAIS shall enforce typing by providing that all operations conform to the type definitions. Every entity, relationship and attribute shall have one and only one type.

4.2B Rules about Type Definitions. The CAIS type definitions shall

- specify the entity types and relationship types to which each attribute type may apply
- specify the type or types of entities that each relationship type may connect and the attribute types allowed for each relatiorship type
- specify the set of allowable elementary values for each attribute type
- specify the relationship types and attribute types for each entity type
- permit relationship types that represent either functional mappings (one-to-one or many-to-one) or relational mappings (one-to-many or many-to-many)
- permit multiple distinct relationships among the same entities
- impose a lattice structure on the types which includes inheritance of attributes, attribute value ranges (possibly restricted), relationships and allowed operations.

4.2C Type Definition. The CAIS shall provide facilities for defining new entity, relationship and attribute types.

4.2D Changing Type Definitions. The CAIS shall provide facilities for changing type definitions. These facilities shall be controlled such that data integrity is maintained.

4.2E Triggering. The CAIS shall provide a conditional triggering mechanism so that prespecified procedures or operations (such as special validation techniques employing multiple attribute value checking) may be invoked whenever values of indicated attributes change. The CAIS shall provide facilities for defining such triggers and the operations or procedures which are tc be invoked.

4.3 Identification The following definitions, used in this subsection, pertain to all the rest of section 4 also:

EXACT IDENTITY

a designation of an entity (or relationship) that is always associated with the entity (or relationship) that it designates. This exact identity will always designate exactly the same entity (or relationship), and it cannot be changed.

IDENTIFICATION

a means of specifying the entities, relationships and attributes to be operated on by a designated operation.

4.3A Exact Identities. The CAIS shall provide exact identities for all entities. The CAIS shall support exact identities for all relationships. The exact identity shall be unique within an instance of a CAIS implementation, and the CAIS shall support a mechanism for the utilization of exact identities across all CAIS implementations.

4.3B Identification. The CAIS shall provide identification of all entities, attributes and relationships. The CAIS shall provide identification of all entities by their exact identity. The CAIS shall support identification of all relationships by their exact identity.

4.3C Identification Methods. The CAIS shall provide identification of entities and relationships by at least the following methods:

- identification of some "start" entity(s), the specification of some relationship type and the specification of some predicate involving attributes or attribute types associated with that relationship type or with some entity type. This method shall identify those entities which are related to the identified start entity(s) by relationships of the given relationship type and for which the predicate is true. Subject to the security constraints of section 2.8, all relationships and entities shall be capable of identification via this method, and all attributes and attribute types (except uninterpreted data) shall be permitted in the predicates.

- identification of an entity type or relationship type and specification of some predicate on the value of any attribute of the entity type or relationship type. This method shall identify those entities or relationships of the given type for which the predicate is true. Subject to the security constraints of section 2.8, all attributes (except uninterpreted data) shall be permitted in the predicates.

4.4 Operations

4.4A Entity Operations. The CAIS shall provide facilities to:

- create entities
- delete entities
- examine entities (by examining their attributes and relationships)
- modify entities (by modifying their attributes)
- identify entities (as specified in Section 4.3)

4.4B Relationship Operations. The CAIS shall provide facilities to:

- create relationships
- delete relationships
- examine relationships (by examining their attributes)
- modify relationships (by modifying their attributes)
- identify relationships (as specified in Section 4.3)

4.4C Attribute Operations. The CAIS shall provide facilities to:

- examine attributes
- modify attributes

4.4D Exact Identity Operations. The CAIS shall provide facilities to:

- pass exact identities between processes
- compare exact identities

4.4E Uninterpreted Data Operations. The CAIS shall provide that use of the input-output facilities of the Ada language (as defined in Chapter 14 of ANSI/MIL-STD-1815A [Ada83]) results in reading/writing an uninterpreted data attribute of an entity. The facilities of Section 6 shall then apply.

4.4F Synchronization. The CAIS shall provide dynamic access synchronization mechanisms to individual entities, relationships and attributes.

4.4G Access Control. The CAIS shall provide selective prohibition of operations on entities, relationships, and attributes being requested by an individual.

4.5 Transaction. The following definition, used in this subsection, pertains to the remainder of section 4 also:

TRANSACTION a grouping of operations, including a designated sequence of operations, which requires that either all of the designated operations are applied or none are; e.g., a transaction is uninterruptible from the user's point of view.

4.5A Transaction Mechanism. The CAIS shall support a transaction mechanism. The effect of running transactions concurrently shall be as if the concurrent transactions were run serially.

4.5B Transaction Control. The CAIS shall support facilities to start, end and abort transactions. When a transaction is aborted, all effects of the designated sequence of operations shall be as if the sequence were never started.

4.5C System Failure. System failure while a transaction is in progress shall cause the effects of the designated sequence of operations to be as if the sequence were never started.

4.6 History. The following definitions, used in this subsection, pertain to the remainder of section 4 also:

HISTORY a recording of the manner in which entities, relationships and attribute values were produced and of all information which was relevant in the production of those entities, relationships or attribute values.

4.6A History Mechanism. The CAIS shall support a mechanism for collecting and utilizing history. The history mechanism shall provide sufficient information to support comprehensive configuration control.

4.7 Robustness and Restoration. The following definitions, used in this subsection, pertain to the remainder of section 4 also:

BACKUP a redundant copy of some subset of the CAIS-managed data. The subset is capable of restoration to active use by a CAIS implementation, particularly in the event of a loss of completeness or integrity in the data in use by implementation.

ARCHIVE a subset of the CAIS-managed data that has been relegated to backing storage media while retaining the integrity, consistency and availability of all information in the entity management system.

4.7A Robustness and Restoration. The CAIS shall support facilities which ensure the robustness of and ability to restore CAIS-managed data. The facilities shall include at least those required to support the backup and archiving capabilities provided by modern operating systems.

5. PROGRAM EXECUTION FACILITIES

Access controls and security rights will apply to all CAIS facilities required by this section.

The following definitions pertain specifically to this section:

PROCESS the CAIS facility used to represent the execution of any program.

PROGRAM a set of compilation units, one of which is a subprogram called the "main program." Execution of the program consists of execution of the main program, which may invoke subprograms declared in the compilation units of the program.

RESOURCE any capacity which must be scheduled, assigned, or controlled by the operating system to assure consistent and non-conflicting usage by programs under execution. Examples of resources include: CPU time, memory space (actuals and virtual), and shared facilities (variables, devices, spoolers, etc.).

ACTIVATE to create a CAIS process. The activation of a program binds that program to its execution environment, which are the resources required to support the process's execution, and includes the program to be executed. The activation of a process marks the earliest point in time which that process can be referenced as an entity within the CAIS environment.

TERMINATE to stop the execution of a process such that it cannot be resumed.

DEACTIVATE to remove a terminated process so that it may no longer be referenced within the CAIS environment.

SUSPEND to stop the execution of a process such that it can resumed. In the context of an Ada program being executed, this implies the suspension of all tasks, and the prevention of the activation of any task until the process is resumed. It specifically does not imply the release of any resources which a process has assigned to it, or which it has acquired, to support its execution.

RESUME to resume the execution of a suspended process.

TASK WAIT delay of the execution of a task within a process until a CAIS service requested by this task has been performed. Other tasks in the same process are not delayed.

MONITOR to observe (or measure) the behavior or value of a process, operation, or data.

5.1 Activation of Program

5.1A Activation. The CAIS shall provide a facility for a process to create a process for a program that has been made ready for execution. This event is called activation.

5.1B Unambiguous Identification. The CAIS shall provide facilities for the unambiguous identification of a process at any time between its activation and deactivation; one such capability shall be as an indivisible part of activation. This act of identification establishes a reference to that process. Once such a reference is established, that reference will refer to the same process until the reference is dissolved. A reference is always dissolved upon termination of the process that established the reference. A terminated process may not be deactivated while there are references to that process.

5.1C Activation Data. The CAIS shall provide a facility to make data available to a program upon its activation.

5.1D Dependent Activation. The CAIS shall provide a facility for the activation of programs that depend upon the activating process for their existence.

5.1E Independent Activation. The CAIS shall provide a facility for the activation of programs that do not depend upon the activating process for their existence.

5.2 Termination

5.2A Termination. The CAIS shall provide a facility for a process to terminate a process. There shall be two forms of termination; the voluntary termination of a process (termed completion) and the abnormal termination of a process. Completion of a process is always self-determined, whereas abnormal termination may be initiated by other processes.

5.2B Termination of Dependent Processes. The CAIS shall support clear, consistent rules defining the termination behavior of processes dependent on a terminating process.

5.2C Termination Data. The CAIS shall provide a facility for termination data to be made available. This data shall provide at least an indication of success or failure for processes that complete. For processes that terminate abnormally the termination data shall indicate abnormal termination.

5.3 Communication

5.3A Data Exchange. The CAIS shall provide a facility for the exchange of data among processes.

5.4 Synchronization

5.4A Task Waiting. The CAIS shall support task waiting.

5.4B Parallel Execution. The CAIS shall provide for the parallel execution of processes.

5.4C Synchronization. The CAIS shall provide a facility for the synchronization of cooperating processes.

5.4D Suspension. The CAIS shall provide a facility for suspending a process.

5.4E Resumption. The CAIS shall provide a facility to resume a process that has been suspended.

5.5 Monitoring

5.5A Identify Reference. The CAIS shall provide a facility for a process to determine an unambiguous identity of a process and to reference that process using that identity.

5.5B RTS Independence. CAIS program execution facilities should be designed to require no additional functionality in the Ada Run-Time System (RTS) from that provided by Ada semantics. Consequently, the implementation of the Ada RTS shall be independent of the CAIS.

5.5C Instrumentation. The CAIS shall provide a facility for a process to inspect and modify the execution environment of another process. This facility is intended to promote support for portable debuggers and other instrumentation tools.

6. INPUT/OUTPUT

Access controls and security rights will apply to all CAIS facilities required by this section.

The requirements specified in this section pertain to input/output between/among processes, data entities, communication devices, and storage devices, unless otherwise stated.

The following definitions pertain specifically to this section:

CONSUMER an entity that is receiving data units via a datapath.

DATA UNIT a representation of a value of an Ada discrete type.

DATAPATH the mechanism by which data units are transmitted from a producer to a consumer.

DEVICE DRIVER
a computer program fragment responsible for converting a device independent information representation or protocol to a device dependent representation or protocol.

PRODUCER an entity that is transmitting data units via a datapath.

TYPE-AHEAD the ability of a producer to transmit data units before the consumer requests the data units

Input and output are defined in terms of device drivers. The following requirements are divided between those required of interfaces to tools, and those required by implementors of device drivers.

6.1 Tool-Device Driver Interfaces

6.1A Specified Interfaces. The CAIS shall specify tool-driver interfaces for at least the following logical devices:

- magnetic tape in ANSI format
- paper tape including precise hole placement

- serial text
- positional text
- graphical write-once
- graphical rewritable
- window manager

6.1B Text Interfaces. The text interfaces shall support control of at least the following attributes of text: margins, page width, page length, boldness, slant, justification, underlining, type size, color, and line spacing. The positional text interface shall permit locator input.

6.1C Graphical Tool Interfaces. The graphical interfaces shall support at least the description by tools of geometrical figures, and of complete pixel-detailed illustrations. There shall be specific interfaces for line drawings in geometrical form, with mechanisms for area fill. There shall be specific interfaces for including text strings in graphical output. The graphical rewritable interface shall permit locator input.

6.1D Device Driver Visibility. The CAIS shall provide interfaces which enable a tool to determine if a device driver is available.

6.1E Unsupported Features. The CAIS shall provide interfaces to control the consequences when the underlying device does not have all of the features required by the device driver.

6.1F Device Driver Connection. The CAIS shall provide interfaces by which a tool can connect to a device driver. This shall permit at least static, and may permit dynamic association between a tool and a device driver.

6.1G Device Driver Creation and Deletion. The CAIS shall provide interfaces which permit the addition and removal of device drivers.

6.1H Data Length. The CAIS shall specify reasonable limits on the length of data items to be communicated across the interfaces it specifies, and shall require all implementations to support to these limits.

6.1I Buffering. The CAIS shall support the clearing of any output buffers, both with and without forced processing of their contents. The CAIS shall support the clearing of input buffers. The CAIS shall support the input of character data such that each character is received when it is transmitted, without waiting for any other condition.

6.1J Data Modifications. The CAIS shall support control of character insertion (padding), character deletion (filtering), and character replacement (modification) in its text interfaces.

6.1K Input Sampling. The CAIS shall support sampling an input device for data without waiting due to an absence of data.

6.1L Type-Ahead. The CAIS shall support device driver interrogation and control of type-ahead.

6.1M Echoing. The CAIS shall support device driver interrogation and control of echoing.

6.1N Control Input Traps. The CAIS shall support the identification of a text device as a control device, i.e. a device for which certain sequences of input data represent a control input trap. The CAIS shall support selection of the sequences and their consequences.

6.1O Telecommunications Support. The CAIS shall support a telecommunications interface for data transmission.

6.2 Interfaces Supporting Device Drivers

6.2A Device Independence. The specifications of the interfaces required by this section shall not be dependent on particular devices.

6.2B Exclusive Access. The CAIS shall provide a means for a device driver to obtain exclusive access to a device, either a physical device or a device driver. Such exclusive access does not require the exclusion of processes which, in a particular installation or implementation, cannot be prevented from intruding.

6.2C Input Output Sequencing. The CAIS shall provide facilities to enable a device driver to ensure the servicing of output requests in the order of their invocation, the processing of input in temporal order, and the proper sequencing of input and output.

6.2D Transmission Characteristics. The CAIS shall support device driver inquiry and control of at least baud rate, parity, number of bits, and full/half duplex.

6.2E Timeout. The CAIS shall provide facilities to permit timeout on input and output operations.

6.2F Data Link Control. The CAIS shall support facilities for the dynamic control of data links, including, at least, self-test, automatic dialing, hang-up, and broken-link handling.

6.3 Common External Form

6.3A Import and Export. The CAIS shall specify a representation on external media of a set of related data entities (as described in section 4), to be known as the Common External Form. The CAIS shall support transferring information from the entity system of Section 4 to external media in this form, and vice versa. The CAIS shall specify that all such transfers preserve at least a) text contents, b) string-valued attributes, and c) relationships to entities transferred at the same time.

RAC Comment Form

!section: !RAC version: **14 July 1986**

!submitter: !date:

!1-line topic/subject:

!extended comment or recommendation:

!rationale for recommendation:

!disposition by RACWG:

[Send via MILNET to *POberndorf@Ada20.ISI.EDU* & *HMumm@Ada20.ISI.EDU*,
or via U.S. Mail to "Patricia Oberndorf/Hans Mumm,
Code 423, NOSC, San Diego, CA 92152"]

RAC / CAIS Version 1

Compliance Study

Prepared for
Naval Ocean Systems Center
San Diego, CA

Under Contract No. N66001-86-D-0156
Delivery Order 0007
CDRL A003AB

Date Prepared: 30 July 1986

Prepared by:
Ada[1] Software Engineering (ASE) Project

TRW

Defense Systems Group
On: Space Park
Redondo Beach, CA 90278

[1] *Ada* is a Registered Trademark of the U.S. Government, Ada Joint Program Office

Table of Contents

SECTION I: EXECUTIVE SUMMARY

1.0 Overview of RAC-CAIS Comparison Analysis Procedures.

An initial traceability analysis was completed to determine those areas in which the proposed Military Standard Common APSE Interface Set, dated 31 January 1985 (referred to as the CAIS Version 1) did not currently address the requirements in the RAC (the DoD Requirements and Design Criteria for the Common APSE Interface Set), dated 13 September 1985. This traceability analysis will be helpful in assessing the amount of effort to anticipate in the generation of CAIS Version 2. This study is primarily statistical, without speculation toward the methodology of the CAIS. The analysis was performed from two perspectives: a detailed analysis of requirements by RAC section, and a more global analysis by categories of key requirements issues deemed most important in the defining of the CAIS.

1.1 Traceability Method.

The results detailed in the following sections were obtained by establishing a standard method for tracing each of the RAC requirements to supporting CAIS Version 1 interfaces.

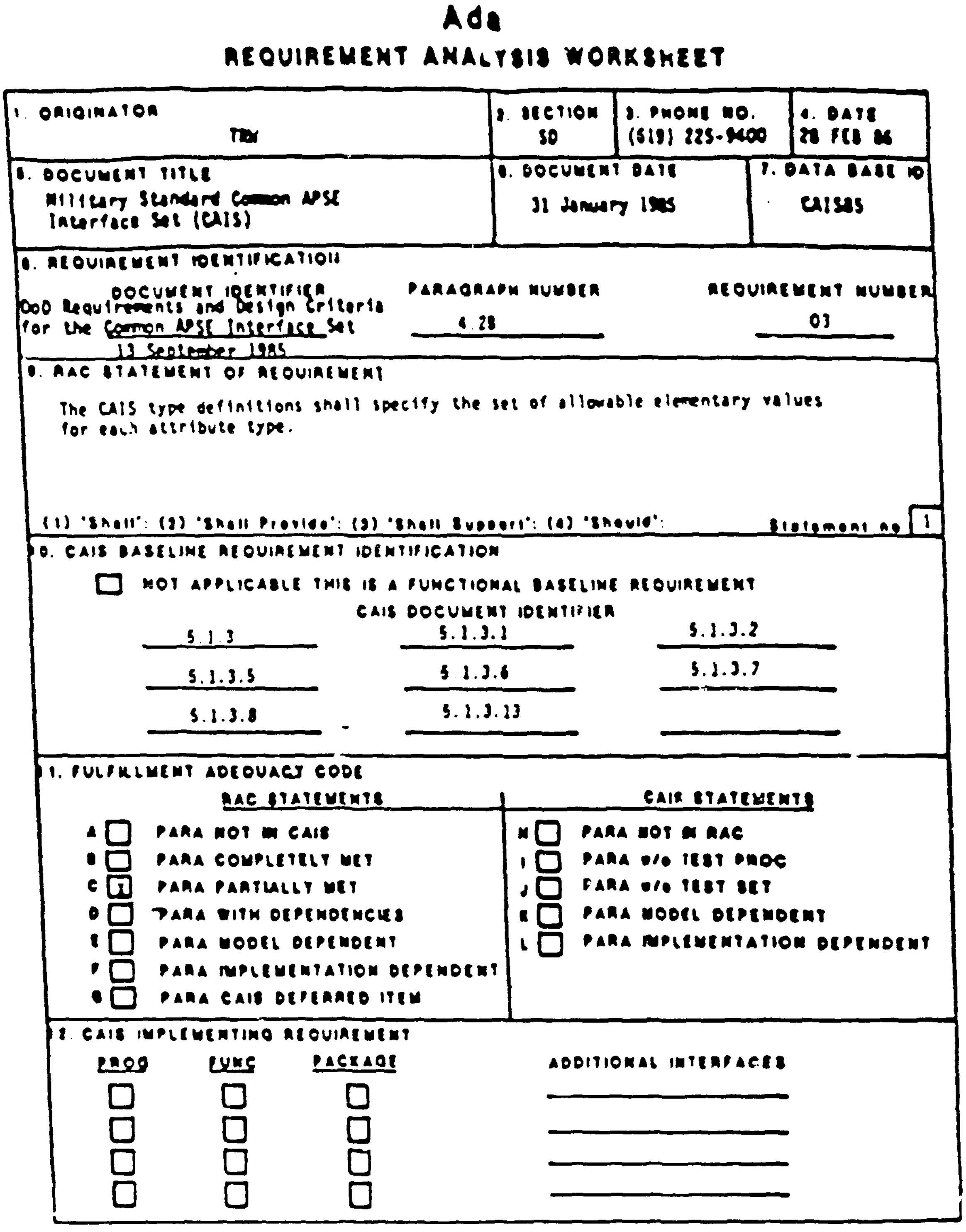
1.1.1 Establishing a Worksheet.

A Worksheet was designed that would allow each RAC requirement to be tracked to its specific CAIS interface(s). An Ada Requirement Analysis Worksheet, Figure 1(a/b), was designed to be used as a standard traceability guideline during the RAC to CAIS Version 1 analysis. Table 1, Ada Requirement Analysis Worksheet Description and Use, explains the Figure 1a worksheet blocks. The Figure 1b worksheet is identical to the Figure 1a worksheet through block 9; the remainder of the worksheet is used to list the name(s) of the CAIS interface(s) that relate to the RAC requirement.

1.1.2 Analysis by RAC Section.

Every numbered RAC requirement paragraph was broken down (decomposed) into individual requirement statements, each containing a single action verb or verb phrase, and each placed on an Ada Requirement Analysis Worksheet labeled by "Paragraph Number" (as used in the RAC) and an appended 2-digit "Requirement Number" to indicate the single-verb requirement statement from the decomposed RAC requirement. Appendix A is a complete listing of the RAC requirements as broken out for this analysis. Figure 1(a/b) uses decomposed requirement 4.2B 03 as an example of a completed Ada Requirement Analysis Worksheet.

Next, each decomposed RAC requirement was traced to a CAIS interface(s) by interpretation of the RAC requirement, assisted by Section 4 (dated 12-19-85), Section 5 (dated 10-3-85), and Section 6 (not dated) of the RAC Rationale document. After all of the CAIS interfaces were associated with RAC requirement(s), each Ada Requirement Analysis Worksheet was then carefully analyzed to determine its Fulfillment Adequacy Code (block 11). A statistical study was completed on each section of the RAC requirements.



Ada

REQUIREMENT ANALYSIS WORKSHEET

1. ORIGINATOR TRW	2. SECTION SD	3. PHONE NO. (619) 225-9400	4. DATE 28 FEB 86
5. DOCUMENT TITLE Military Standard Common APSE Interface Set (CAIS)	6. DOCUMENT DATE 31 January 1985	7. DATA BASE ID CAIS85	

8. REQUIREMENT IDENTIFICATION

DOCUMENT IDENTIFIER	PARAGRAPH NUMBER	REQUIREMENT NUMBER
DoD Requirements and Design Criteria for the Common APSE Interface Set 13 September 1985	4.28	03

9. RAC STATEMENT OF REQUIREMENT

The CAIS type definitions shall specify the set of allowable elementary values for each attribute type.

(1) 'Shall'; (2) 'Shall Provide'; (3) 'Shall Support'; (4) 'Should'; Statement no 1

10. CAIS BASELINE REQUIREMENT IDENTIFICATION

□ NOT APPLICABLE THIS IS A FUNCTIONAL BASELINE REQUIREMENT

CAIS DOCUMENT IDENTIFIER

5.1.3	5.1.3.1	5.1.3.2
5.1.3.5	5.1.3.6	5.1.3.7
5.1.3.8	5.1.3.13	

11. FULFILLMENT ADEQUACY CODE

RAC STATEMENTS

- A □ PARA NOT IN CAIS
- B □ PARA COMPLETELY MET
- C ☒ PARA PARTIALLY MET
- D □ PARA WITH DEPENDENCIES
- E □ PARA MODEL DEPENDENT
- F □ PARA IMPLEMENTATION DEPENDENT
- G □ PARA CAIS DEFERRED ITEM

CAIS STATEMENTS

- H □ PARA NOT IN RAC
- I □ PARA w/o TEST PROC
- J □ PARA w/o TEST SET
- K □ PARA MODEL DEPENDENT
- L □ PARA IMPLEMENTATION DEPENDENT

12. CAIS IMPLEMENTING REQUIREMENT

PROG	FUNC	PACKAGE	ADDITIONAL INTERFACES
□	□	□	
□	□	□	
□	□	□	
□	□	□	

Figure 1a

Ada
REQUIREMENT ANALYSIS WORKSHEET

1. ORIGINATOR TRW	2. SECTION SD	3. PHONE NO. (619)225-9400	4. DATE 28 FEB 86
5. DOCUMENT TITLE Military Standard Common APSE Interface Set (CAIS)	6. DOCUMENT DATE 31 January 1985		7. DATA BASE ID CAIS 85

8. REQUIREMENT IDENTIFICATION

DOCUMENT IDENTIFIER	PARAGRAPH NUMBER	REQUIREMENT NUMBER
DoD Requirements and Design Criteria for the Common APSE Interface Set 13 September 1985	4.2B	03

9. RAC STATEMENT OF REQUIREMENT

The CAIS type definitions shall specify the set of allowable elementary values for each attribute type.

(1) 'Shall': (2) 'Shall Provide': (3) 'Shall Support': (4) 'Should'. Statement no 1

CAIS INTERFACE(S)

5.1.3 Package Attributes

5.1.3.1 Create Node Attribute
Value in List Type (Initial Value)

5.1.3.2 Create Path Attribute
Value in List Type (Initial Value)

5.1.3.5 Set Node Attribute
Value in List Type (New Value of Attribute)

5.1.3.6 Set Path Attribute
Value in List Type (New Value of Attribute)

5.1.3.7 Get Node Attribute
Value in List Type (Result Parameter Containing the Value of Attribute)

5.1.3.8 Get Path Attribute
Value in List Type (Result Parameter Containing the Value of Attribute)

5.1.3.13 Get Next
Value in List Type (Result Parameter Containing the Value of the Attribute Named by Attribute).

Figure 1b

TABLE 1

Ada Requirement Analysis Worksheet Description and Use

Worksheet Block No.	Description and use
1.	Originator. Name of the company preparing the Requirement Analysis Worksheet. (TRW)
2.	Section. Identification of the group of which the analyst is a member. (SD)
3.	Phone No. The Phone number of the analyst within the group preparing the worksheet. (619 225-9400)
4.	Date. The date that the analysis was completed. (28 February 1986)
5.	Document Title Title of document that the traceability analysis study was performed on. (Military Standard Common APSE Interface Set (CAIS) Version 1).
6.	Document Date. The date of release of the document being analyzed as shown on the cover of the document (31 January 1985).
7.	Document Data Base ID. This entry is an alphanumeric mnemonic database identifier to be used with the Ada Traceability tool for comparing one CAIS Version analysis to another (CAIS85).

TABLE 1 (Continued)

Ada Requirement Analysis Worksheet Description and Use

Worksheet Block No. — Description and use

8. Requirement Identification.
This entry is composed of three components as follows:

Document Identifier. Title of requirements document that the traceability analysis study was performed against. (DoD Requirements and Design Criteria for the Common APSE Interface Set (CAIS), dated 13 September 1985 (RAC))

Paragraph Number. The exact paragraph number from which the requirement is extracted.

Requirement Number. This is a two digit number from 01 to 99 that uniquely identifies a specific requirement in those paragraphs that contain more than one requirement.

Example: RAC requirement 4.3B stated:
" The CAIS shall provide identification of all entities, attributes, and relationships."

This statement was broken down to requirement numbers listed in block 8 of the Ada Requirements Analysis Worksheets as:

4.3B 01 (The CAIS shall provide identification of all entities.)
4.3B 02 (The CAIS shall provide identification of all relationships.)
4.3B 03 (The CAIS shall provide identification of all attributes.)

9. RAC Statement of Requirement.
This is a complete sentence(s) in subject-verb-object format that states the requirement. Only one requirement is stated per worksheet. Block 9 also tracks the verb or verb phrases that indicate where and to what degree individual constraints apply. Each requirement will have one of the following verbs or verb phrases assigned by a coded number. These verbs or verb phrases with their coded number are:

(1) "SHALL" - indicates a requirement on the definiti [illegible] ne CAIS.

(2) "SHALL PROVIDE" - indicates a requirement for the CAIS to provide interface(s) with prescribed capabilities.

(3) "SHALL SUPPORT" - indicates a requirement for the CAIS to provide interface(s) with prescribed capabilities or for CAIS definers to demonstrate that the capability may be constructed from CAIS interfaces.

(4) "SHOULD" - indicates a desired goal but one for which there is no objective test

TABLE 1 (Continued)

Ada Requirement Analysis Worksheet Description and Use

Worksheet Block No. Description and use

10. CAIS Baseline Requirement Identification.
The CAIS interface(s) paragraph number that trace directly to the RAC Statement of Requirement (block 9) were listed here. If more than nine CAIS interfaces were required, a Block 10 only worksheet was attached. The "NOT APPLICABLE THIS IS A FUNCTIONAL BASELINF REQUIREMENT" Block is "X'ed" if the RAC Statement of Requirement pertained to all or many CAIS interfaces, e.g., RAC requirement 3.1A 02 "The Syntax of the CAIS shall conform to the character set as defined by the Ada standard."

11. Fulfillment Adequacy Code.
The analyst has "X'ed" the square that represents the analyst's assessment of how well the CAIS Baseline interface(s) (block 10) fulfills the intent of the RAC requirement statement (block 9).

These assignments were based on the following guidelines and conditions.

A. PARA NOT IN CAIS - The RAC requirement is not a deferred item and the requirement is not represented in this specific CAIS Version or statements/interfaces in the CAIS were so limited as to not be truly representative of the requirement.

B. PARA SUBSTANTIALLY MET - The RAC requirement was largely or completely satisfied within the limits of this specific CAIS version, or the relevant CAIS interface was clearly designed to completely satisfy the RAC requirement, but it is not yet possible to determine whether it will.

C. PARA PARTIALLY MET - The RAC requirement was represented in the CAIS version but was not complete or consistent throughout the CAIS version.

D. PARA WITH DEPENDENCIES - (Not used in this analysis)

E. PARA MODEL DEPENDENT - (Not used in this analysis)

F. PARA IMPLEMENTATION DEPENDENT - (Not used in this analysis)

G. PARA DEFERRED ITEM - The RAC requirement has been formally deferred for this specific CAIS version. That is, it has been intentionally excluded from the scope of the CAIS Version 1, and recognized as needed in Version 2 of the CAIS.

12 CAIS Implementing Requirement
This Block is for future use and was not used in this analysis

1.2 Key Issues Perspective

Several key requirements concepts were abstracted from the RAC as the "most important" categories of properties that the CAIS is required to fulfill. The detailed RAC/CAIS analysis (described in Section 1.1 of this study) was supplemented by this "Key Issues" analysis in lieu of applying weighting factors in deriving the detailed scores. Because the relative importance of various detailed requirements varies greatly, some accommodation is necessary in the RAC/CAIS analysis to offset any misleading conclusions that might be inferred from the unweighted statistics. Because universally satisfactory detailed weighting factors would be difficult to quantify, this "most important categories" approach was chosen as a way of focusing the results in terms of issues that can be viewed abstractly, thus providing a more equitable perspective on CAIS fulfillment of its global purposes.

In support of this analysis, the RAC requirements have also been broken down into the key issues categories. The RAC requirements were analysed to determine whether they address one or more of these key issues, and if so, which. Section 3 of this study is a brief discussion of the key issues considered, and Appendix B contains the detailed tables indicating which RAC requirements address which issues, and the degree to which the requirements are met by the CAIS, according to the analysis in the body of this document.

Some issues are represented by many requirements in the RAC, and some by a few; the number of requirements did not always reflect the importance of the issue. Thus, counting the total number of requirements satisfied in the RAC as a whole is not an indication of the degree to which these key issues are covered. Secondly, for several of the key issues there are some critical requirements that are not fully covered in the RAC. Thus, noting that most of the RAC requirements relating to a given issue are satisfied may not lead to an accurate conclusion with regard to the functionality of the CAIS in that area.

A good example of this is the issue of Distribution. Most of the requirements in the RAC regarding distribution have to do with implementing an exact identity that would be constant across different CAIS instances. The CAIS Version 1 satisfies some of these requirements, but was not intended to support distribution, whereas another interface set might support distribution, but implement it without using exact identities. In such a situation, looking at the number of RAC requirements satisfied would lead one to an incorrect conclusion about the interface sets' relative support for distribution.

In addition, the requirements relevant to a given issue are not necessarily allocated evenly to the various subfunctions that fall under that heading. Therefore, the number of RAC requirements relevant to a given key issue that are satisfied by the CAIS may not accurately reflect the degree of coverage of some particular subfunction in that category. Moreover, some requirements are subordinate to others. These would need to be aggregated in order to assess a more realistic degree of compliance to the RAC.

For example, the RAC contains a large number of requirements regarding Entity Management Support. The CAIS fulfills many of these, but a few particular requirements, like typing, make a significant difference in the overall power of the CAIS interfaces. This would be much more clearly shown if the requirements were grouped and evaluated by major functional areas within each category.

Ideally, the requirements within each issue, as well as the issues themselves, would be weighted, and some quantitative measure of RAC satisfaction would be determined. However, the importance of any given issue is context dependent, and may often be evaluated for non-technical purposes. Clearly, different issues may be identified at any given time as more important for a particular use of the CAIS.

Therefore, the approach taken here is to assign RAC requirements to one or more of these key issues, and then indicate how many of the requirements relevant to each issue are satisfied by CAIS Version 1. The subjective weighting of this data is left to the user of this document, who may assign relative weights to each issue, and to the various requirements relevant to each issue, depending on the needs of his environment.

SECTION II: TRACEABILITY METHOD

2.0 Introduction to Traceability Method.

The Section II paragraphs with their associated tables contain a complete and detailed description of the analytic results which resulted from the Traceability Method of RAC-CAIS comparison described in Section 1.1 and which led to the conclusions provided in Section 4.1 and Section 5.

2.1 Study by Sections.

Tables 3 through 7 reflect the analysis of the RAC Requirements by Section. Each table is broken down by Fulfillment Adequacy Code, Percent of Fulfillment within its Section, and Action Verb or Verb Phrase. (See Appendix A for a complete list of RAC requirement statements and assigned requirement numbers.)

2.1.1 RAC Section 2.

RAC Section 2, General Design Objectives, was broken down into 19 (9.4%) of the 203 RAC requirements. Table 2, RAC Section 2, is a complete analysis of the General Design Objectives section of the RAC Requirements Document. Figure 2 is a summary of the Section 2 analysis.

```
   0% of Section 2 RAC Requirements are assigned Fulfillment Code A.
94.7% of Section 2 RAC Requirements are assigned Fulfillment Code B.
 5.3% of Section 2 RAC Requirements are assigned Fulfillment Code C.
   0% of Section 2 RAC Requirements are assigned Fulfillment Code G.
```

Figure 2 Section 2 Analysis in Summary

CAIS Version 1 substantially satisfies 94.7% of the requirements set forth in the RAC Section 2, while 5.3% of the RAC requirements are partially met.

2.1.2 RAC Section 8.

RAC Section 3, General Syntax and Semantics, was broken down into 22 (10.8%) of the 203 RAC requirements. Table 3, RAC Section 3, is a complete analysis of the General Syntax and Semantics section of the RAC Requirements Document. Figure 3 is a summary of the Section 3 analysis.

```
   0% of Section 3 RAC Requirements are assigned Fulfillment Code A.
68.2% of Section 3 RAC Requirements are assigned Fulfillment Code B.
31.8% of Section 3 RAC Requirements are assigned Fulfillment Code C.
   0% of Section 3 RAC Requirements are assigned Fulfillment Code G.
```

Figure 3 Section 3 Analysis in Summary

CAIS Version 1 substantially satisfies 68.2% of the requirements set forth in the RAC Section 3 while 31.8% of the RAC requirements are partially met.

2.1.3 RAC Section 4.

RAC Section 4, Entity Management Support, was broken down into 65 (32.0%) of the 203 RAC requirements. Table 4, RAC Section 4, is a complete analysis of the Entity Management Support section of the RAC Requirements Document. Figure 4 is a summary of the Section 4 analysis.

```
18 5% of Section 4 RAC Requirements are assigned Fulfillment Code A.
50 7% of Section 4 RAC Requirements are assigned Fulfillment Code B.
12.3% of Section 4 RAC Requirements are assigned Fulfillment Code C.
18 5% of Section 4 RAC Requirements are assigned Fulfillment Code G.
```

Figure 4 Section 4 Analysis in Summary

CAIS Version 1 substantially satisfies 50.7% of the requirements set forth in the RAC Section 4 while 12.3% of the RAC requirements are partially met. 18.5% are deferred items and 18.5% are not represented in the CAIS Version 1.

2.1.4 RAC Section 5.

RAC Section 5, Program Execution Facilities, was broken down into 21 (10.3%) of the 203 RAC requirements. Table 5, RAC Section 5, is a complete analysis of the Program Execution Facilities section of the RAC Requirements Document. Figure 5 is a summary of the Section 5 analysis.

```
 9 5% of Section 5 RAC Requirements are assigned Fulfillment Code A.
81.0% of Section 5 RAC Requirements are assigned Fulfillment Code B.
 9 5% of Section 5 RAC Requirements are assigned Fulfillment Code C.
 0.0% of Section 5 RAC Requirements are assigned Fulfillment Code G.
```

Figure 5 Section 5 Analysis in Summary

CAIS Version 1 substantially satisfies 81.0% of the requirements set forth in the RAC Section 5 while 9.5% of the RAC requirements are partially met. 9.5% of the RAC requirements are not represented in the CAIS Version 1 document.

2.1.5 RAC Section 6.

RAC Section 6, Input/Output, was broken down into 76 (37.4%) of the 203 RAC requirements. Table 6, RAC Section 6, is a complete analysis of the Input/Output section of the RAC Requirements Document. Figure 6 is a summary of the Section 6 analysis.

```
 0 0% of Section 6 RAC Requirements are assigned Fulfillment Code A.
21.0% of Section 6 RAC Requirements are assigned Fulfillment Code B.
14 5% of Section 6 RAC Requirements are assigned Fulfillment Code C.
64 5% of Section 6 RAC Requirements are assigned Fulfillment Code G.
```

Figure 6 Section 6 Analysis in Summary

CAIS Version 1 substantially satisfies 21.0% of the requirements set forth in the RAC Section 6 while 14.5% of the RAC requirements are partially met. 64.5% of the RAC requirements are deferred items in the CAIS Version 1 document.

2.2 Study by Fulfillment Adequacy Code.

Tables 8 through 12 reflect the analysis of the RAC Requirements by Fulfillment Adequacy Code. (See Appendix A for a complete list of RAC Requirement Statements and assigned requirement numbers.)

2.2.1 Fulfillment Adequacy Code A (PARA NOT IN CAIS).

Table 7 shows a complete list of RAC Requirements that are assigned Fulfillment Adequacy Code A. 6.9% of the RAC Requirements are not included in CAIS Version 1.

2.2.2 Fulfillment Adequacy Code B (PARA SUBSTANTIALLY MET).

Table 8 shows a complete list of RAC Requirements that are assigned Fulfillment Adequacy Code B. 48.8% of the RAC Requirements are substantially met in CAIS Version 1.

2.2.3 Fulfillment Adequacy Code C (PARA PARTIALLY MET).

Table 9 shows a complete list of RAC Requirements that are assigned Fulfillment Adequacy Code C. 14.5% of the RAC Requirements are partially met in CAIS Version 1.

2.2.4 Fulfillment Adequacy Code G (PARA DEFERRED ITEM).

Table 10 shows a complete list of RAC Requirements that are assigned Fulfillment Adequacy Code G. 30% of the RAC Requirements are deferred items in CAIS Version 1.

2.3 Requirement by Action Verb or Verb Phrases.

Table 11 lists each RAC requirement by its assigned action verb or verb phrase. This table is supplied so that the user may compare the action verbs "SHALL, SHALL PROVIDE, SHALL SUPPORT, SHOULD" with present or future planning of CAIS revisions or updates.

TABLE 2 RAC SECTION 2

GENERAL DESIGN OBJECTIVES

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code **	Percent of Fulfillment
2.1	01	2	B	
2.2	01	4	B	
2.3	01	1	B	
2.3	02	1	B	
2.3	03	1	B	
2.3	04	1	B	
2.3	06	4	B	
2.4	01	4	B	
2.5	01	4	B	94.7% (18/19)
2.6	01	1	B	
2.7	01	4	B	
2.7	02	4	B	
2.7	03	4	B	
2.7	04	4	B	
2.8	01	2	B	
2.8	02	1	B	
2.8	03	3	B	
2.8	04	1	B	
2.3	05	1	C	5.3% (1/19)

There are 19 Section 2 requirements (see Appendix A for breakdown)

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

** Fulfillment Code Key:
A - NOT IN CAIS
B - SUBSTANTIALLY MET
C - PARTIALLY MET
G - DEFERRED ITEM

TABLE 3 RAC SECTION 3

GENERAL SYNTAX AND SEMANTICS

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code **	Percent of Fulfillment
3.1A	01	1	B	
3.1A	02	1	B	
3.1B	01	4	B	
3.1C	01	4	B	
3.1C	02	4	B	
3.1C	03	4	B	
3.1C	04	4	B	68.2% (15/22)
3.1C	05	4	B	
3.1D	01	4	B	
3.2A	02	4	B	
3.2C	01	1	B	
3.2C	02	1	B	
3.2C	03	1	B	
3.2E	01	4	B	
3.2F	01	1	B	
3.1B	02	4	C	
3.2A	01	1	C	
3.2A	03	1	C	
3.2B	01	2	C	31.8% (7/22)
3.2B	02	1	C	
3.2B	03	4	C	
3.2D	01	4	C	

There are 22 Section 3 requirements (see Appendix A for breakdown)

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

** Fulfillment Code Key:
A - NOT IN CAIS
B - SUBSTANTIALLY MET
C - PARTIALLY MET
G - DEFER' ITEM

TABLE 4 RAC SECTION 4

ENTITY MANAGEMENT SUPPORT

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code **	Percent of Fulfillment
4.2E	01	2	A	
4.2E	02	2	A	
4.3A	01	2	A	
4.3A	02	3	A	
4.3A	03	1	A	
4.3A	04	3	A	18.5% (12/65)
4.3B	04	2	A	
4.3B	05	3	A	
4.3C	01	2	A	
4.3C	02	2	A	
4.4D	01	2	A	
4.4D	02	2	A	
4.1A	01	2	B	
4.1A	02	2	B	
4.1A	03	2	B	
4.1C	01	1	B	
4.2B	01	1	B	
4.2B	02	1	B	
4.2B	03	1	B	
4.2B	04	1	B	
4.2B	05	1	B	
4.2B	07	1	B	
4.3B	01	2	B	
4.3B	02	2	B	
4.3B	03	2	B	
4.4A	01	2	B	
4.4A	02	2	B	
4.4A	03	2	B	50.7% (33/65)
4.4A	04	2	B	
4.4B	01	2	B	
4.4B	02	2	B	
4.4B	03	2	B	
4.4B	04	2	B	
4.4C	01	2	B	
4.4C	02	2	B	
4.4E	01	2	B	
4.4E	02	2	B	
4.4F	01	2	B	
4.4F	02	2	B	
4.4F	03	2	B	
4.4G	01	2	B	
4.4G	02	2	B	
4.4G	03	2	B	
4.6A	01	3	B	
4.6A	02	2	B	

TABLE 4 RAC SECTION 4 (Cont')

ENTITY MANAGEMENT SUPPORT

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code **	
4.1B	01	2	C	
4.2A	01	1	C	
4.4A	05	2	C	
4.4B	05	2	C	12.3% (8/65)
4.6B	01	3	C	
4.7A	01	3	C	
4.7A	02	3	C	
4.7A	03	3	C	
4.2B	06	1	G	
4.2B	08	1	G	
4.2C	01	2	G	
4.2C	02	2	G	
4.2C	03	2	G	
4.2D	01	2	G	18.5% (12/65)
4.2D	02	1	G	
4.5A	01	3	G	
4.5B	01	3	G	
4.5B	02	3	G	
4.5B	03	3	G	
4.5C	01	1	G	

There are 65 Section 4 requirements (see Appendix A for breakdown)

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

** Fulfillment Code Key:
A - NOT IN CAIS
B - SUBSTANTIALLY MET
C - PARTIALLY MET
G - DEFERRED ITEM

TABLE 5 RAC SECTION 5

PROGRAM EXECUTION FACILITIES

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code **	Percent of Fulfillment
5.5C	01	2	A	
5.5C	02	2	A	9.5% (2/21)
5.1A	01	2	B	
5.1B	01	2	B	
5.1C	01	2	B	
5.1D	01	2	B	
5.1E	01	2	B	
5.2A	01	2	B	
5.2A	02	2	B	
5.2B	01	3	B	
5.2C	01	2	B	81.0% (17/21)
5.2C	02	2	B	
5.2C	03	1	B	
5.3A	01	2	B	
5.4B	01	2	B	
5.4C	01	2	B	
5.4D	01	2	B	
5.4E	01	2	B	
5.5A	01	2	B	
5.4A	01	3	C	
5.5B	01	1	C	9.5% (2/21)

There are 21 Section 5 requirements (see Appendix A for breakdown)

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

** Fulfillment Code Key:
A - NOT IN CAIS
B - SUBSTANTIALLY MET
C - PARTIALLY MET
G - DEFERRED ITEM

TABLE 6 RAC SECTION 6

INPUT/OUTPUT

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code **	Percent of Fulfillment
6.1A	01	2	B	
6.1B	01	2	B	
6.2A	01	2	B	
6.2B	01	2	B	
6.3A	01	2	B	
6.3B	01	2	B	
6.3B	02	2	B	
6.3C	01	2	B	
6.3C	02	2	B	
6.3F	01	2	B	21.0% (16/76)
6.3F	02	2	B	
6.4J	01	2	B	
6.4J	02	2	B	
6.4J	03	2	B	
6.4J	04	1	B	
6.4J	05	1	B	
6.3D	01	2	C	
6.3D	02	2	C	
6.3D	03	2	C	
6.3D	04	2	C	
6.3H	01	2	C	14.5% (11/76)
6.3H	02	2	C	
6.4B	01	2	C	
6.4B	02	2	C	
6.7A	01	1	C	
6.7A	02	1	C	
6.7B	01	4	C	
6.1C	01	2	G	
6.1C	02	2	G	
6.1D	01	2	G	
6.1E	01	3	G	
6.1F	01	3	G	
6.3E	01	2	G	
6.3E	02	2	G	
6.3G	01	2	G	
6.4A	01	2	G	
6.4A	02	2	G	
6.4A	03	2	G	
6.4C	01	2	G	

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

** Fulfillment Code Key:
A - NOT IN CAIS
B - SUBSTANTIALLY MET
C - PARTIALLY MET
G - DEFERRED ITEM

TABLE 6 RAC SECTION 6 (Cont')

INPUT/OUTPUT

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code **	
6.4C	02	2	G	
6.4D	01	1	G	
6.4D	02	1	G	
6.4D	03	2	G	
6.4D	04	2	G	
6.4E	01	1	G	
6.4E	02	1	G	
6.4E	03	2	G	
6.4E	04	2	G	
6.4F	01	1	G	
6.4F	02	1	G	
6.4F	03	2	G	
6.4F	04	2	G	
6.4G	01	2	G	
6.4H	01	3	G	
6.4I	01	2	G	64.5% (49/76)
6.4I	02	2	G	
6.4I	03	2	G	
6.4I	04	1	G	
6.4I	05	1	G	
6.4K	01	2	G	
6.4L	01	2	G	
6.4L	02	2	G	
6.4M	01	2	G	
6.4M	02	2	G	
6.4M	03	2	G	
6.4M	04	2	G	
6.4M	05	2	G	
6.4M	06	2	G	
6.4N	01	3	G	
6.4N	02	3	G	
6.4N	03	3	G	
6.4N	04	3	G	
6.5A	01	2	G	
6.6A	01	1	G	
6.6B	01	2	G	
6.6B	02	2	G	

There are 76 Section 6 requirements (see Appendix A for breakdown)

* Action Verb Key:	** Fulfillment Code Key:
(1) "SHALL"	A - NOT IN CAIS
(2) "SHALL PROVIDE"	B - SUBSTANTIALLY MET
(3) "SHALL SUPPORT"	C - PARTIALLY MET
(4) "SHOULD"	G - DEFERRED ITEM

TABLE 7 FULFILLMENT CODE A

PARA NOT IN CAIS

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code
4 2E	01	2	A
4.2E	02	2	A
4.3A	01	2	A
4.3A	02	3	A
4.3A	03	1	A
4.3A	04	3	A
4.3B	04	2	A
4.3B	05	3	A
4.3C	01	2	A
4.3C	02	2	A
4.4D	01	2	A
4.4D	02	2	A
5.5C	01	2	A
5.5C	02	2	A

FULFILLMENT CODE A = 14 OF 203 REQUIREMENTS

THIS REPRESENTS 6.9% OF THE RAC REQUIREMENTS FOR THIS SPECIFIC CAIS VERSION

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

TABLE 8 FULFILLMENT CODE B

PARA SUBSTANTIALLY MET

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code
2.1	01	2	B
2.2	01	4	B
2.3	01	1	B
2.3	02	1	B
2.3	03	1	B
2.3	04	1	B
2.3	06	4	B
2.4	01	4	B
2.5	01	4	B
2.6	01	1	B
2.7	01	4	B
2.7	02	4	B
2.7	03	4	B
2.7	04	4	B
2.8	01	2	B
2.8	02	1	B
2.8	03	3	B
2.8	04	1	B
3.1A	01	1	B
3.1A	02	1	B
3.1B	01	4	B
3.1C	01	4	B
3.1C	02	4	B
3.1C	03	4	B
3.1C	04	4	B
3.1C	05	4	B
3.1D	01	4	B
3.2A	02	4	B
3.2C	01	1	B
3.2C	02	1	B
3.2C	03	1	B
3.2E	01	4	B
3.2F	01	1	B

TABLE 8 FULFILLMENT CODE B (Cont')

PARA SUBSTANTIALLY MET

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code
4.1A	01	2	B
4.1A	02	2	B
4.1A	03	2	B
4.1C	01	1	B
4.2B	01	1	B
4.2B	02	1	B
4.2B	03	1	B
4.2B	04	1	B
4.2B	05	1	B
4.2B	07	1	B
4.3B	01	2	B
4.3B	02	2	B
4.3B	03	2	B
4.4A	01	2	B
4.4A	02	2	B
4.4A	03	2	B
4.4A	04	2	B
4.4B	01	2	B
4.4B	02	2	B
4.4B	03	2	B
4.4B	04	2	B
4.4C	01	2	B
4.4C	02	2	B
4.4E	01	2	B
4.4E	02	2	B
4.4F	01	2	B
4.4F	02	2	B
4.4F	03	2	B
4.4G	01	2	B
4.4G	02	2	B
4.4G	03	2	B
4.6A	01	3	B
4.6A	02	2	B

TABLE 8 FULFILLMENT CODE B (Cont')

PARA SUBSTANTIALLY MET

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code
5.1A	01	2	B
5.1B	01	2	B
5.1C	01	2	B
5.1D	01	2	B
5.1E	01	2	B
5.2A	01	2	B
5.2A	02	2	B
5.2B	01	3	B
5.2C	01	2	B
5.2C	02	2	B
5.2C	03	1	B
5.3A	01	2	B
5.4B	01	2	B
5.4C	01	2	B
5.4D	01	2	B
5.4E	01	2	B
5.5A	01	2	B
6.1A	01	2	B
6.1B	01	2	B
6.2A	01	2	B
6.2B	01	2	B
6.3A	01	2	B
6.3B	01	2	B
6.3B	02	2	B
6.3C	01	2	B
6.3C	02	2	B
6.3F	01	2	B
6.3F	02	2	B
6.4J	01	2	B
6.4J	02	2	B
6.4J	03	2	B
6.4J	04	1	B
6.4J	05	1	B

FULFILLMENT CODE B = 99 OF 203 REQUIREMENTS

THIS REPRESENTS 48.8% OF THE RAC REQUIREMENTS FOR THIS SPECIFIC CAIS VERSION

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

TABLE 9 FULFILLMENT CODE C

PARA PARTIALLY MET

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code
2.3	05	1	C
3.1B	02	4	C
3.2A	01	1	C
3.2A	02	1	C
3.2B	01	2	C
3.2B	02	1	C
3.2B	03	4	C
3.2D	01	4	C
4.1B	01	2	C
4.2A	01	1	C
4.4A	05	2	C
4.4B	05	2	C
4.6B	01	3	C
4.7A	01	3	C
4.7A	02	3	C
4.7A	03	3	C
5.4A	01	3	C
5.5B	01	1	C
6.3D	01	2	C
6.3D	02	2	C
6.3D	03	2	C
6.3D	04	2	C
6.3H	01	2	C
6.3H	02	2	C
6.4B	01	2	C
6.4B	02	2	C
6.7A	01	1	C
6.7A	02	1	C
6.7B	01	4	C

FULFILLMENT CODE C = 29 OF 203 REQUIREMENTS

THIS REPRESENTS 14.3% OF THE RAC REQUIREMENTS FOR THIS SPECIFIC CAIS VERSION

TABLE 10 FULFILLMENT CODE G

PARA DEFERRED ITEM

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code
4.2B	06	1	G
4.2B	08	1	G
4.2C	01	2	G
4.2C	02	2	G
4.2C	03	2	G
4.2D	01	2	G
4.2D	02	1	G
4.5A	01	3	G
4.5B	01	3	G
4.5B	02	3	G
4.5B	03	3	G
4.5C	01	1	G
6.1C	01	2	G
6.1C	02	2	G
6.1D	01	2	G
6.1E	01	3	G
6.1F	01	3	G
6.3E	01	2	G
6.3E	02	2	G
6.3G	01	2	G
6.4A	01	2	G
6.4A	02	2	G
6.4A	03	2	G
6.4C	02	2	G
6.4C	01	2	G
6.4D	01	1	G
6.4D	02	1	G
6.4D	03	2	G
6.4D	04	2	G
6.4E	01	1	G
6.4E	02	1	G
6.4E	03	2	G
6.4E	04	2	G
6.4F	01	1	G
6.4F	02	1	G
6.4F	03	2	G
6.4F	04	2	G
6.4G	01	2	G
6.4H	01	3	G

TABLE 10 FULFILLMENT CODE G (cont.)

PARA DEFERRED ITEM

RAC paragraph number	Rqmnt number	Action verb *	Fulfillment code
6.4I	01	2	G
6.4I	02	2	G
6.4I	03	2	G
6.4I	04	1	G
6.4I	05	1	G
6.4K	01	2	G
6.4L	01	2	G
6.4L	02	2	G
6.4M	01	2	C
6.4M	02	2	G
6.4M	03	2	G
6.4M	04	2	G
6.4M	05	2	G
6.4M	06	2	G
6.4N	01	3	G
6.4N	02	3	G
6.4N	03	3	G
6.4N	04	3	G
6.5A	01	2	G
6.6A	01	1	G
6.6B	01	2	G
6.6B	02	2	G

FULFILLMENT CODE G = 61 OF 203 REQUIREMENTS
THIS REPRESENTS 30% OF THE RAC REQUIREMENTS FOR THIS SPECIFIC CAIS VERSION

* Action Verb Key:
(1) "SHALL"
(2) "SHALL PROVIDE"
(3) "SHALL SUPPORT"
(4) "SHOULD"

TABLE 11: REQUIREMENTS BY ASSIGNED VERB PHRASE

ACTION VERB "SH[illegible]

RAC rqmnt	Rqmnt ID	.ction verb	Fulfillment code *
4 3A	03	1	A
2 3	01	1	B
2 3	02	1	B
2 3	03	1	B
2 3	04	1	B
2 6	01	1	B
2 8	02	1	B
2 8	04	1	B
3 1A	01	1	B
3 1A	02	1	B
3 2C	01	1	B
3 2C	02	1	B
3 2C	03	1	B
3 2F	01	1	B
4 1C	01	1	B
4 2B	01	1	B
4 2B	02	1	B
4 2B	03	1	B
4 2B	04	1	B
4 2B	05	1	B
4 2B	07	1	B
5 2C	03	1	B
6 4J	04	1	B
6 4J	05	1	B

* Fulfillment Code Key

B - PARA SUBSTANTIALLY MET	A - PARA NOT IN CAIS
C - PARA PARTIALLY MET	G - PARA DEFERRED ITEM

TABLE 11. REQUIREMENTS BY ASSIGNED VERB PHRASE (cont.)

ACTION VERB "SHALL" (cont.)

RAC rqmnt	Rqmnt ID	Action verb	Fulfillment code *
2.3	05	1	C
3.2A	01	1	C
3.2A	03	1	C
3.2B	02	1	C
4.2A	01	1	C
5.5B	01	1	C
6.7A	01	1	C
6.7A	02	1	C
4.2B	06	1	G
4.2B	06	1	G
4.2D	02	1	G
4.5C	01	1	G
6.4D	01	1	G
6.4D	02	1	G
6.4E	01	1	G
6.4E	02	1	G
6.4F	01	1	G
6.4F	02	1	G
6.4I	04	1	G
6.4I	05	1	G
6.6A	01	1	G

* Fulfillment Code Key:

B - PARA SUBSTANTIALLY MET	A - PARA NOT IN CAIS
C - PARA PARTIALLY MET	G - PARA DEFERRED ITEM

TABLE 11: REQUIREMENTS BY ASSIGNED VERB PHRASE (cont.)

ACTION VERB "SHALL PROVIDE"

RAC rqmnt	Rqmnt ID	Action verb	Fulfillment code *
4.2E	01	2	A
4.2E	02	2	A
4.3A	01	2	A
4.3B	04	2	A
4.3C	01	2	A
4.3C	02	2	A
4.4D	01	2	A
4.4D	02	2	A
5.5C	01	2	A
5.5C	02	2	A
2.1	01	2	B
2.8	01	2	B
4.1A	01	2	B
4.1A	02	2	B
4.1A	03	2	B
4.3B	01	2	B
4.3B	02	2	B
4.3B	03	2	B
4.4A	01	2	B
4.4A	02	2	B
4.4A	03	2	B
4.4A	04	2	B
4.4B	01	2	B
4.4B	02	2	B
4.4B	03	2	B
4.4B	04	2	B
4.4C	01	2	B
4.4C	02	2	B
4.4E	01	2	B
4.4E	02	2	B
4.4F	01	2	B
4.4F	02	2	B
4.4F	03	2	B
4.4G	01	2	B
4.4G	02	2	B
4.4G	03	2	B
4.6A	02	2	B

* Fulfillment Code Key

B - PARA SUBSTANTIALLY MET   A - PARA NOT IN CAIS
C - PARA PARTIALLY MET   G - PARA DEFERRED ITEM

TABLE 11: REQUIREMENTS BY ASSIGNED VERB PHRASE (cont.)

ACTION VERB "SHALL PROVIDE" (cont.)

RAC rqmnt	Rqmnt ID	Action verb	Fulfillment code *
5.1A	01	2	B
5.1B	01	2	B
5.1C	01	2	B
5.1D	01	2	B
5.1E	01	2	B
5.2A	01	2	B
5.2A	02	2	B
5.2C	01	2	B
5.2C	02	2	B
5.3A	01	2	B
5.4B	01	2	B
5.4C	01	2	B
5.4D	01	2	B
5.4E	01	2	B
5.5A	01	2	B
6.1A	01	2	B
6.1B	01	2	B
6.2A	01	2	B
6.2B	01	2	B
6.3A	01	2	B
6.3B	01	2	B
6.3B	02	2	B
6.3C	01	2	B
6.3C	02	2	B
6.3F	01	2	B
6.3F	02	2	B
6.4J	01	2	B
6.4J	02	2	B
6.4J	03	2	B

* Fulfillment Code Key:

B - PARA SUBSTANTIALLY MET	A - PARA NOT IN CAIS
C - PARA PAPTIALLY MET	G - PARA DEFERRED ITEM

TABLE 11. REQUIREMENTS BY ASSIGNED VERB PHRASE (cont.)

ACTION VERB "SHALL PROVIDE" (cont.)

RAC rqmnt	Rqmnt ID	Action verb	Fulfillment code *
3.2B	01	2	C
4.1B	01	2	C
4.4A	05	2	C
4.4B	05	2	C
6.3D	01	2	C
6.3D	02	2	C
6.3D	03	2	C
6.3D	04	2	C
6.3H	01	2	C
6.3H	02	2	C
6.4B	01	2	C
6.4B	02	2	C

* Fulfillment Code Key:

B - PARA SUBSTANTIALLY MET A - PARA NOT IN CAIS

C - PARA PARTIALLY MET G - PARA DEFERRED ITEM

TABLE 11: REQUIREMENTS BY ASSIGNED VERB PHRASE (cont.)

ACTION VERB "SHALL PROVIDE" (cont.)

RAC rqmnt	Rqmnt ID	Action verb	Fulfillment code *
4.2C	01	2	G
4.2C	02	2	G
4.2C	03	2	G
4.2D	01	2	G
6.1C	01	2	G
6.1C	02	2	G
6.1D	01	2	G
6.3E	01	2	G
6.3E	02	2	G
6.3G	01	2	G
6.4A	01	2	G
6.4A	02	2	G
6.4A	03	2	G
6.4C	01	2	G
6.4C	02	2	G
6.4D	03	2	G
6.4D	04	2	G
6.4E	03	2	G
6.4E	04	2	G
6.4F	03	2	G
6.4F	04	2	G
6.4G	01	2	G
6.4I	01	2	G
6.4I	02	2	G
6.4I	03	2	G
6.4K	01	2	G
6.4L	01	2	G
6.4L	02	2	G
6.4M	01	2	G
6.4M	02	2	G
6.4M	03	2	G
6.4M	04	2	G
6.4M	05	2	G
6.4M	06	2	G
6.5A	01	2	G
6.6B	01	2	G
6.6B	02	2	G

Fulfillment Code Key:
A - PARA NOT IN CAIS
B - PARA SUBSTANTIALLY MET
C - PARA PARTIALLY MET
G - PARA DEFERRED ITEM

TABLE 11: REQUIREMENTS BY ASSIGNED VERB PHRASE (cont.)

ACTION VERB "SHALL SUPPORT"

RAC rqmnt	Rqmnt ID	Action verb	Fulfillment code *
4 3A	02	3	A
4 3A	04	3	A
4.3B	05	3	A
2.8	03	3	B
4.6A	01	3	B
5.2B	01	3	B
4 6B	01	3	C
4.7A	01	3	C
4.7A	02	3	C
4.7A	03	3	C
5.4A	01	3	C
4.5A	01	3	G
4.5B	01	3	G
4.5B	02	3	G
4.5B	03	3	G
6.1E	01	3	G
6.1F	01	3	G
6.4H	01	3	G
6.4N	01	3	G
6.4N	02	3	G
6 4N	03	3	G
6.4N	04	3	G

* Fulfillment Code Key:

B - PARA SUBSTANTIALLY MET	A - PARA NOT IN CAIS
C - PARA PARTIALLY MET	G - PARA DEFERRED ITEM

TABLE 11: REQUIREMENTS BY ASSIGNED VERB PHRASE (cont.)

ACTION VERB "SHOULD"

RAC rqmnt	Rqmnt ID	Action verb	Fulfillment code *
2.2	01	4	B
2.3	06	4	B
2.4	01	4	B
2.5	01	4	B
2.7	01	4	B
2.7	02	4	B
2.7	03	4	B
2.7	04	4	B
3.1B	01	4	B
3.1C	01	4	B
3.1C	02	4	B
3.1C	03	4	B
3.1C	04	4	B
3.1C	05	4	B
3.1D	01	4	B
3.2A	02	4	B
3.2E	01	4	B
3.1B	02	4	C
3.2B	03	4	C
3.2D	01	4	C
6.7B	01	4	C

* Fulfillment Code Key:

B - PARA SUBSTANTIALLY MET	A - PARA NOT IN CAIS
C - PARA PARTIALLY MET	G - PARA DEFERRED ITEM

SECTION III: KEY ISSUES METHOD

3.0 Overview of Key Issues Analysis

The critical areas of functionality described in the RAC are discussed in this section. Appendix B contains tables showing the assignment of RAC requirements to specific key issues, with the evaluations assigned each requirement in the detailed study by section. There is no implied ordering in the list of key issues; many of the requirements are interrelated. Each category is described below, along with the number of RAC requirements applicable to each category, and the fulfillment codes applicable to these requirements.

Key Issue 1: Support for Ada

The CAIS was intended as an interface set to support the KIT/KITIA goal of commonality among Ada Programming Support Environments (APSEs). This issue concerns support for Ada as the language for the interface and for the implementation of the facilities.

There are 24 RAC requirements relating to Support for Ada.

75.0% (18/24) are Substantially Met (B)
25.0% (6/24) are Partially Met (C)

Key Issue 2: Widely Implementable

The CAIS specification is required to be machine-independent to support the commonality described above, so that it can be implemented on commonly used software development systems (hardware and/or operating systems).

There are 11 RAC requirements relating to Wide Implementability

63.6% (7/11) are Substantially Met (B)
36.4% (4/11) are Partially Met (C)

Key Issue 3: Support for Security

Security is of increasing importance in any environment that will support software development for the Department of Defense or the Services. An APSE interface set must support the required security policies.

There are 9 RAC requirements relating to Support for Security

100.0% (9/ 9) are Substantially Met (B)

Key Issue 4: Input/Output

Commonality and wide implementability require that the CAIS support a wide variety of facilities for input/output among objects (e.g., processes, data entities, communication devices, and storage devices). The breadth and richness of functionality supported are important measures of the usefulness of an APSE interface set.

There are 79 RAC requirements relating to Input/Output

22.8% (18/79) are Substantially Met (B)
15.2% (12/79) are Partially Met (C)
62.0% (49/79) are Deferred Items (G)

Key Issue 5: Entity Management Support

The database is the repository for the information that enables an APSE to support an integrated toolset. Appropriate support for the management of this information is critical to the functionality of any APSE interface set. Specific references for critical areas of functionality include the following:

a. Typing
b. Transactions
c. Identification
d. Operations
e. Triggering

There are 67 RAC requirements relating to Entity Management Support

17.9% (12/67) are Not In CAIS (A)
52.2% (35/67) are Substantially Met (B)
12.0% (8/67) are Partially Met (C)
17.9% (12/67) are Deferred Items (G)

Key Issue 6: Program Execution Facilities

Tools using the APSE execute as processes that require support from the interface set. Appropriate support for the management of executing processes and their related information is critical for the functionality of an APSE interface set. Note that some of these areas are related to functions discussed in other categories. Specific references for critical areas of functionality include the following:

a. Communication
b. Synchronization
   (1) Task Waiting
   (2) Transactions
   (3) Data Object Locking
   (4) Message, Queues
   (5) Process Creation and Termination
c. Monitoring

There are 33 RAC requirements relating to Program Execution Facilities

```
 6.1% ( 2/33) are Not In CAIS    (A)
66.7% (22/33) are Substantially Met (B)
12.1% ( 4/33) are Partially Met (C)
15.1% ( 5/33) are Deferred Items (G)
```

Key Issue 7: Distribution

APSEs are expected to support the software development environments that will be commonly used in the near future. Distributed systems are playing an increasingly important role in these environments, and thus are a necessary feature of an APSE interface set. (There are no explicit references in the RAC to a distributed environment, although many subsections assume this requirement. There is currently a proposal to add a specific requirement for Distribution to the RAC.) RAC requirements in support of a distributed environment include those relating to Exact Identity.

There are 8 RAC requirements relating to Distribution

```
100.0% ( 8/ 8) are Not In CAIS    (A)
```

Several other categories are also considered to be critical requirements for a CAIS to satisfy, but they are much less well-defined than the other categories, so it is more difficult to quantify the extent of compliance. These categories are given here.

Key Issue 8: Level of the Interfaces

A CAIS should contain interfaces that provide functionality at a level appropriate for the needs of the tools that use it.

There are 3 RAC requirements relating to the Level of the Interfaces

```
100 0% ( 3/ 3) are Substantially Met (B)
```

Key Issue 9: Appropriateness of the Interfaces

The interfaces of a CAIS must provide the support necessary for the tools that use it. The interface set must be complete, adequate, and appropriate for the needs of the full APSEs and software engineering environments of the foreseeable future.

There are 4 RAC requirements relating to the Appropriateness of the Interfaces

```
75.0% ( 3/ 4) are Substantially Met (B)
25.0% ( 1/ 4) are Partially Met (C)
```

Key Issue 10: Technology Compatibility and Evolution

A CAIS should reflect the current state-of-the-practice in support services for software development tools, but must be able to evolve as the practice evolves.

There are 4 RAC requirements relating to Technology Compatibility and Evolution

75.0% (3/ 4) are Substantially Met (B)
25.0% (1/ 4) are Partially Met (C)

SECTION IV: SUMMARY

4.0 Summary of Analyses.

4.1 Analysis by RAC Sections.

The following table (Table 12) represents a summary of the Requirements Study by RAC sections. A complete analysis of the results for each section is given in Section 2 of this study.

TABLE 12
Summary of Requirements Study

	RAC Sections					
Fulfillment Codes *	2	3	4	5	6	All
A	0.0%	0.0%	18.5%	9.5%	0.0%	6.9% (14)
B	94.7%	68.2%	50.7%	81.0%	21.0%	48.8% (99)
C	5.3%	31.8%	12.3%	9.5%	14.5%	14.3% (29)
G	0.0%	0.0%	18.5%	0.0%	64.5%	30.0% (61)
	100%	100%	100%	100%	100%	100%
# of reqmnts per section	(19)	(22)	(65)	(21)	(76)	(203)

* A - PARA NOT IN CAIS
B - PARA SUBSTANTIALLY MET
C - PARA PARTIALLY MET
G - PARA DEFERRED ITEM

4.2 Analysis by Key Issues.

The following table represents a summary of the Key Issue Study.

Summary of Key Issue Study

	Fulfillment Codes *				
Key Issue Number	A	B	C	G	
1	0.0%	75.0%	25.0%	0.0%	(24/203)
2	0.0%	63.6%	36.4%	0.0%	(11/203)
3	0.0%	100.0%	0.0%	0.0%	(9/203)
4	0.0%	22.8%	15.2%	62.0%	(79/203)
5	17.9%	52.2%	12.0%	17.9%	(67/203)
6	6.1%	66.7%	12.1%	15.1%	(33/203)
7	100.0%	0.0%	0.0%	0.0%	(8/203)
8	0.0%	100.0%	0.0%	0.0%	(3/203)
9	0.0%	75.0%	25.0%	0.0%	(4/203)
10	0.0%	75.0%	25.0%	0.0%	(4/203)

* Fulfillment Codes:

A - PARA NOT IN CAIS
B - PARA SUBSTANTIALLY MET
C - PARA PARTIALLY MET
G - PARA DEFERRED ITEM

SECTION V: CONCLUSIONS

5. Conclusions.

Statistically, via the traceability method, CAIS Version 1 substantially or partially addresses 63.1% of the requirements set forth in the RAC. 30% of the RAC requirements are formally deferred items and 6.9% of the RAC requirements are neither fulfilled nor stated as formally deferred in the CAIS Version 1 document.

From the Key Issue perspective, those RAC requirements that define the Entity Management Support and Program Execution Facilities are largely met. In Entity Management Support, the only RAC requirements not met in the CAIS are those relating to Typing, Triggering, Transactions, and Exact Identity. In Program Execution Facilities, only Transactions and Instrumentation are not supported. Essential functionality in all these areas, as well as in Input/Output, exists in CAIS Version 1. Distribution has been formally deferred for this CAIS Version, and many subfunctions of Input/Output have also been deferred. In all the other Key Issue areas, the CAIS Version 1 substantially or partially satisfies all the RAC Requirements.

This study reinforces a prevalent general impression that the CAIS needs a lot of work in the I/O area. It also shows that CAIS Version 1, while designed before the RAC was written, nevertheless satisfies much of the RAC requirements, with specific exceptions primarily in areas that were called out by the CAIS specifiers as Deferred Topics.

APPENDIX A: Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmt No.	Section 2 RAC Statements
2.1	01	The CAIS shall provide interfaces sufficient to support the use of APSEs for wide classes of projects throughout their lifecycles and to promote I & T among APSEs.
2.2	01	The CAIS should provide simple-to-use mechanisms for achieving common simple actions. [Features which support needs of less frequently used tool should be given secondary consideration.]
2.3	01	The CAIS specification shall be machine independent
2.3	02	The CAIS specification shall be implementation independent
2.3	03	The CAIS shall be implementable on bare machines.
2.3	04	The CAIS shall be implementable on machines with any of a variety of operating systems.
2.3	05	The CAIS shall contain only interfaces which provide facilities which have been demonstrated in existing commercial or military software systems.
2.3	06	CAIS features should be chosen to have a simple and efficient implementation in many machines, to avoid execution cost for unneeded generality and to ensure that unused portions of a CAIS implementation will not add to execution of a non-using tool. [The measures of the efficiency criterion are, primarily, minimum interactive response time APSE tools and, secondarily, consumption of resources.]
2.4	01	Interfaces should be partitioned such that the partitions may be understood independently and they contain no undocumented dependencies between partitions.
2.5	01	The design of the CAIS should facilitate development and use of extensions of the CAIS. [i.e. CAIS interfaces should be reusable so that they can be combined to create new interfaces and facilities.]
2.6	01	The CAIS shall adopt existing standards where applicable. [For example, recognized standards for device characteristics are provided by ANSI, ISO, IEEE, and DoD.]

[ ] is used for criterion or to amplify the requirement.

Appendix A (continued)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 2 RAC Statements (Cont')
2.7	01	All CAIS features should uniformly address aspects such as status returns.
2.7	02	All CAIS features should uniformly address aspects such as exceptional conditions.
2.7	03	All CAIS features should uniformly address aspects such as parameter types.
2.7	04	All CAIS features should uniformly address aspects such as options.
2.8	01	The CAIS shall provide interfaces to allow tools to operate within a Trusted Computer System [TCS] that meet the Class B3 criteria as defined in [TCSEC83].
2.8	02	It shall be possible to implement the CAIS within a TCS.
2.8	03	When implemented within a TCS, the CAIS shall support the use of the security facilities provided by the Trusted Computing Base [TCB] to applications programs.
2.8	04	When not implemented within a TCS, the CAIS interfaces sensitive to security shall operate as a dedicated secure system [i.e., all data at a single security level, and all subjects cleared to at least that level].

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 3 RAC Statements
3.1A	01	The syntax of the CAIS shall be expressed as Ada package specifications.
3.1A	02	The syntax of the CAIS shall conform to the character set as defined by the Ada standard [section 2.1 of ANSI/mil-STD-1815A [Ada83]].
3.1B	01	The CAIS should employ uniform syntactic conventions and should not provide several notations for the same concept.
3.1B	02	CAIS syntax issues [including, at least, limits on name length, abbreviation styles, other naming conventions, relative ordering of input and output parameters, etc] should be resolved in a uniform and integrated manner for the whole CAIS.
3.1C	01	The CAIS should avoid coining new words [literals or identifiers].
3.1C	02	The CAIS should avoid using words in an unconventional sense.
3.1C	03	Ada identifiers [names] defined by the CAIS should be natural language words or industry accepted terms whenever possible.
3.1C	04	The CAIS should define Ada identifiers which are visually distinct and not easily confused [including, at least, that the CAIS should avoid defining two Ada identifiers that are only a 2 character transposition away from being identical].
3.1C	05	The CAIS should use the same name everywhere in the interface set, and not its possible synonyms, when the same meaning is intended.
3.1D	01	The CAIS should impose only those restrictive rules or constraints required to achieve I&T. [CAIS implementors will be required to provide the complete specifications of all syntactic restrictions imposed by their CAIS implementations.]

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 3 RAC Statements (cont.)
3.2A	01	The CAIS shall be completely and unambiguously defined.
3.2A	02	The specification of semantics should be both precise and understandable.
3.2A	03	The semantic specification of each CAIS interface shall include a precise statement of assumptions [including execution-time preconditions for calls], effects on global data and packages, and interactions with other interfaces.
3.2B	01	The CAIS shall provide responses for all interface calls, including informative non-null responses [return value or exception] for unsuccessful completions.
3.2B	02	All responses returned across CAIS interfaces shall be defined in an implementation independent manner.
3.2B	03	Every time a CAIS interface is called under the same circumstances, it should return the same response.
3.2C	01	The CAIS interfaces shall employ the mechanism of Ada exceptions to report exceptional situations that arise in the execution of CAIS facilities.
3.2C	02	The CAIS specification shall include exceptions [with visible declarations] for all situations that violate the preconditions specified for the CAIS interfaces.
3.2C	03	The CAIS specification shall include exceptions [with visible declarations] that cover all violations of implementation-defined restrictions.
3.2D	01	The description of CAIS semantics should use the same word or phrase everywhere, and not its synonyms, when the same meaning is intended.
3.2E	01	Each CAIS interface should provide only one function.
3.2F	01	The CAIS specification shall enumerate all aspects of the meanings of CAIS interfaces and facilities which must be defined by CAIS implementors. [CAIS implementors will be required to provide the complete specifications for these implementation-defined semantics.]

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 4 RAC Statements
4.1A	01	The CAIS shall provide facilities for representing data using entities.
4.1A	02	The CAIS shall provide facilities for representing data using attributes.
4.1A	03	The CAIS shall provide facilities for representing data using binary relationships.
4.1B	01	The CAIS shall provide facilities for representing data as elementary values.
4.1C	01	The CAIS shall ensure the integrity of the CAIS-managed data.
4.2A	01	The facilities provided by the CAIS shall enforce typing by providing that all operations conform to the type definitions.
4.2B	01	The CAIS type definitions shall specify the entity types and relationship types to which each attribute type may apply.
4.2B	02	The CAIS type definitions shall specify the type or types of entities that each relationship type may connect and the attribute types allowed for each relationship type.
4.2B	03	The CAIS type definitions shall specify the set of allowable elementary values for each attribute type.
4.2B	04	The CAIS type definitions shall specify the relationship types and attributes types for each entity type.
4.2B	05	The CAIS type definitions shall permit relationship types that represent functional mappings [one-to-one or many-to-one].
4.2B	06	The CAIS type definitions shall permit relationship types that represent relational mappings [one-to-many or many-to-many].
4.2B	07	The CAIS type definitions shall permit multiple distinct relationships among the same entities.
4.2B	08	The CAIS type definitions shall impose a lattice structure on the types which includes inheritance of attributes, attribute value ranges [possibly restricted], relationships and allowed operations.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 4 RAC Statements (cont.)
4.2C	01	The CAIS shall provide facilities for defining new entity types.
4.2C	02	The CAIS shall provide facilities for defining new relationship types.
4.2C	03	The CAIS shall provide facilities for defining new attribute types.
4.2D	01	The CAIS shall provide facilities for changing type definitions.
4.2D	02	The facilities for changing type definitions shall be controlled such that data integrity is maintained.
4.2E	01	The CAIS shall provide a conditional triggering mechanism so that prespecified procedures or operations [i.e. such as special validation techniques employing multiple attribute value checking] may be invoked whenever values of indicated attributes change.
4.2E	02	The CAIS shall provide facilities for defining such triggers and the operations or procedures [i.e. such as special validation techniques employing multiple attribute value checking] which are to be invoked.
4.3A	01	The CAIS shall provide exact identities for all entities.
4.3A	02	The CAIS shall support exact identities for all relationships.
4.3A	03	The exact identity shall be unique within an instance of a CAIS implementation.
4.3A	04	The CAIS shall support a mechanism for the utilization of exact identities across all CAIS implementations.
4.3B	01	The CAIS shall provide identification of all entities.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)
Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 4 RAC Statements (cont.)
4.3B	02	The CAIS shall provide identification of all attributes.
4.3B	03	The CAIS shall provide identification of all relationships.
4.3B	04	The CAIS shall provide identification of all entities by their exact identity.
4.3B	05	The CAIS shall support identification of all relationships by their exact identity.
4.3C	01	The CAIS shall provide identification of entities [by at least the following methods: 1. Identification of some "start" entity specification of some predicate on the value of any attribute of the entity type. 2. Identification of an entity type and specification of some predicate on the value of any attribute of the entity type.]
4.3C	02	The CAIS shall provide identification of relationships [by at least the following method: 1. The specification of some relationship type. 2. Identification of a relationship type and specification of some predicate on the value of any attribute of the relationship type.]
4.4A	01	The CAIS shall provide facilities to create entities.
4.4A	02	The CAIS shall provide facilities to delete entities.
4.4A	03	The CAIS shall provide facilities to examine entities [by examining their attributes and relationships].
4.4A	04	The CAIS shall provide facilities to modify entities [by modifying their attributes].
4.4A	05	The CAIS shall provide facilities to identify entities [as specified in Section 4.3].
4.4B	01	The CAIS shall provide facilities to create relationships.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)
Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 4 RAC Statements (cont.)
4.4B	02	The CAIS shall provide facilities to delete relationships.
4.4B	03	The CAIS shall provide facilities to examine relationships [by examining their attributes].
4.4B	04	The CAIS shall provide facilities to modify relationships [by modifying their attributes].
4.4B	05	The CAIS shall provide facilities to identify relationships [as specified in Section 4.3].
4.4C	01	The CAIS shall provide facilities to examine attributes.
4.4C	02	The CAIS shall provide facilities to modify attributes.
4.4D	01	The CAIS shall provide facilities to pass exact identities between processes.
4.4D	02	The CAIS shall provide facilities to compare exact identities.
4.4E	01	The CAIS shall provide that use of the input-output facilities of the Ada Language [as defined in Chapter 4 of the ANSI/MIL-STD-1815A [Ada83]] results in reading an uninterpreted data attribute or an entity. [The facilities of Section 6 shall then apply.]
4.4E	02	The CAIS shall provide that use of the input-output facilities of the Ada language [as defined in Chapter 14 of ANSI/MIL-STD-1815A [Ada83]] results in writing an uninterpreted data attribute of an entity. [The facilities of Section 6 shall then apply.]
4.4F	01	The CAIS shall provide dynamic access synchronization mechanisms to individual entities.
4.4F	02	The CAIS shall provide dynamic access synchronization mechanisms to individual relationships.
4.4F	03	The CAIS shall provide dynamic access synchronization mechanisms to individual attributes.
4.4G	01	The CAIS shall provide selective prohibition of operations on entities being requested by an individual.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)
Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 4 RAC Statements (cont.)
4.4G	02	The CAIS shall provide selective prohibition of operations on relationships being requested by an individual.
4.4G	03	The CAIS shall provide selective prohibition of operations on attributes being requested by an individual.
4.5A	01	The CAIS shall support a transaction mechanism. [The effect of running transactions concurrently shall be as if the concurrent transactions were run serially.]
4.5B	01	The CAIS shall support facilities to start transactions.
4.5B	02	The CAIS shall support facilities to end transactions.
4.5B	03	The CAIS shall support the facilities to abort transactions. [When a transaction is aborted, all effects of the designated sequence of operations shall be as if the sequence were never started.]
4.5C	01	System failure while a transaction is in progress shall cause the effects of the designated sequence of operations to be as if the sequence were never started.
4.6A	01	The CAIS shall support a mechanism for collecting and utilizing history.
4.6A	02	The history mechanism shall provide sufficient information to support comprehensive configuration control.
4.6B	01	The CAIS shall support mechanisms for ensuring the fidelity of the history.
4.7A	01	The CAIS shall support facilities which ensure the robustness of CAIS-managed data.
4.7A	02	The CAIS shall support facilities which ensure the ability to restore CAIS-managed data. [The facilities shall include at least those required to support the backup capabilities provided by modern operating systems.]
4.7A	03	The CAIS shall support facilities which ensure the ability to restore CAIS-managed data. [The facilities shall include at least those required to support the archiving capabilities provided by modern operating systems.]

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)
Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 5 RAC Statements
5.1A	01	The CAIS shall provide a facility for a process to create a process for a program that has been made ready for execution. [This event is called activation.]
5.1B	01	The CAIS shall provide facilities for the unambiguous identification of a process at any time between its activation and deactivation.
5.1C	01	The CAIS shall provide a facility to make data available to a program upon its activation.
5.1D	01	The CAIS shall provide a facility for the activation of programs that depend upon the activating process for their existence.
5.1E	01	The CAIS shall provide a facility for the activation of programs that do not depend upon the activating process for their existence.
5.2A	01	The CAIS shall provide a facility for a process to terminate a process by voluntary termination of a process [termed completion. Completion of a process is always self-determined.]
5.2A	02	The CAIS shall provide a facility for a process to terminate a process by abnormal termination of a process. [Abnormal termination may be initiated by other processes.]
5.2B	01	The CAIS shall support clear, consistent rules defining the termination behavior of processes dependent on a terminating process.
5.2C	01	The CAIS shall provide a facility for termination data to be made available. This data shall provide an indication of success for processes that complete.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)
Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 5 RAC Statements (cont.)
5.2C	02	The CAIS shall provide a facility for termination data to be made available. [This data shall provide an indication of failure for processes that complete.]
5.2C	03	For processes that terminate abnormally, the termination data shall indicate abnormal termination.
5.3A	01	The CAIS shall provide a facility for the exchange of data among processes.
5.4A	01	The CAIS shall support task waiting.
5.4B	01	The CAIS shall provide for the parallel execution of processes.
5.4C	01	THe CAIS shall provide a facility for the synchronization of cooperating processes.
5.4D	01	The CAIS shall provide a facility for suspending a process.
5.4E	01	The CAIS shall provide a facility to resume a process that has been suspended.
5.5A	01	The CAIS shall provide a facility for a process to determine an unambiguous identity of a process and to reference that process using that identity.
5.5B	01	CAIS program execution facilities shall be designed to require no additional functionality in the Ada Run-Time System [RTS] from that provided by Ada semantics. [Consequently, the implementation of the Ada RTS shall be independent of the CAIS.]
5.5C	01	The CAIS shall provide a facility for a process to inspect the execution environment of another process.
5.5C	02	The CAIS shall provide a facility for a process to modify the execution environment of another process.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 6 RAC Statements
6.1A	01	The CAIS shall provide interfaces for the control of hardcopy terminals.
6.1B	01	The CAIS shall provide interfaces for the control of page terminals
6.1C	01	The CAIS shall provide interfaces for the control of character-imaging printers.
6.1C	02	The CAIS shall provide interfaces for the control of bit-map printers.
6.1D	01	The CAIS shall provide interfaces for the control of paper tape drives.
6.1E	01	The CAIS shall support the control of interactive graphical input/output devices.
6.1F	01	The CAIS shall support a telecommunications interface for data transmissions.
6.2A	01	The CAIS shall provide interfaces for the control of character-imaging block terminals.
6.2B	01	The CAIS shall provide interfaces for the control of magnetic tape drives.
6.3A	01	The datapath control facilities of the CAIS shall be provided at a level comparable to that of Ada Reference Manual's File I/O. That is, control of datapaths shall be provided via subprogram calls rather that via the data units transmitted to the device.
6.3B	01	The CAIS shall provide facilities to permit timeout on input.
6.3B	02	The CAIS shall provide facilities to permit timeout on output operations.
6.3C	01	The CAIS shall provide facilities to obtain exclusive access to a producer; such exclusive access does not prevent a privileged process from transmitting to the consumer.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 6 RAC Statements (cont.)
6.3C	02	The CAIS shall provide facilities to obtain exclusive access to a consumer.
6.3D	01	The CAIS shall provide facilities to associate at execution time the producer of each input datastream with a specific device, data entity, or process.
6.3D	02	The CAIS shall provide facilities to associate at execution time the producer of each output datastream with a specific device, data entity, or process.
6.3D	03	The CAIS shall provide facilities to associate at execution time the consumer of each input datastream with a specific device, data entity, or process.
6.3D	04	The CAIS shall provide facilities to associate at execution time the consumer of each output datastream with a specific device, data entity, or process.
6.3E	01	The CAIS shall provide facilities for the specification of the sizes of input datapath buffers during process execution.
6.3E	02	The CAIS shall provide facilities for the specification of the sizes of output datapath buffers during process execution.
6.3F	01	The CAIS shall provide facilities for the removal of all buffered data from an input datapath.
6.3F	02	The CAIS shall provide facilities for the removal of all buffered data from an output datapath.
6.3G	01	The CAIS shall provide facilities to force the output of all data in an output datapath.
6.3H	01	The CAIS shall provide facilities to ensure the servicing of input requests in the order of their invocation.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 6 RAC Statements (cont.)
6 3H	02	The CAIS shall provide facilities to ensure the servicing of output requests in the order of their invocation.
6.4A	01	The CAIS shall provide input/output facilities for communication with devices requiring 5-bit data units.
6 4A	02	The CAIS shall provide input/output facilities for communication with devices requiring 7-bit data units.
6 4A	03	The CAIS shall provide input/output facilities for communication with devices requiring 8-bit data units.
6.4B	01	The CAIS shall provide the ability to transmit data units and sequences of units without modification.
6 4B	02	The CAIS shall provide the ability to receive data units and sequences of units without modification.
6 4C	01	The CAIS shall provide facilities for the input of single data units. [The completion of this operation makes the data unit available to its consumer(s) without requiring another input/output event, including the receipt of a termination or escape sequence, the filling of a buffer, or the invocation of an operation to force input/output.]
6 4C	02	The CAIS shall provide facilities for the output of single data units. [The completion of this operation makes the data unit available to its consumer[s] without requiring another input/output event, including the receipt of a termination or escape sequence, the filling of a buffer, or the invocation of an operation to force input/output.]
6 4D	01	The CAIS shall specify the set of data units and sequences of units [including the null set] which can be added to an input datastream.

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 6 RAC Statements (cont)
6.4D	02	The CAIS shall specify the set of data units and sequences of units [including the null set] which can be added to an output datastream.
6.4D	03	The CAIS shall provide facilities permitting a process to select at execution time the subset of data units and sequences of units which may be added [including the null set]
6.4D	04	The CAIS shall provide facilities permitting a process to query at execution time the subset of data units and sequences of which may be added [including the null set].
6.4E	01	The CAIS shall specify the set of data units and sequences of units [including the null set] which may be filtered from an input datastream.
6.4E	02	The CAIS shall specify the set of data units and sequences of units [including the null set] which may be filtered from an output datastream.
6.4E	03	The CAIS shall provide facilities permitting a process to select at execution time the subset of data units and sequences of units which may be filtered [including the null set].
6.4E	04	The CAIS shall provide facilities permitting a process to query at execution time the subset of data units and sequences of units which may be filtered [including the null set].
6.4F	01	The CAIS shall specify the set of modifications that can occur to data units in an input datastream [e g , mapping from lower case to upper case]
6 4F	02	The CAIS shall specify the set of modifications that can occur to data units in an output datastream [e g , mapping from lower case to upper case].
6 4F	03	The CAIS shall provide facilities permitting a process to select at execution time the subset of modifications that may occur [including the null set].
6 4F	04	The CAIS shall provide facilities permitting a process to query at execution time the subset of modifications that may occur [including the null set].

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No	Section 6 RAC Statements (cont.)
6.4G	01	The CAIS shall provide facilities to sample an input datapath for available data without having to wait if data is not available.
6.4H	01	The CAIS shall support control at execution time of host transmission characteristics [e.g., rates, parity, number of bits, half/full duplex].
6.4I	01	The CAIS shall provide facilities to disable type-ahead.
6.4I	02	The CAIS shall provide facilities to enable type-ahead.
6.4I	03	The CAIS shall provide facilities to indicate whether type-ahead is supported in the given implementation.
6.4I	04	The CAIS shall define the results of invoking the facilities to disable type-ahead in those implementations that do not support type-ahead. [e.g., null-effect or exception raised].
6.4I	05	The CAIS shall define the results of invoking the facilities to enable type-ahead in those implementations that do not support type-ahead [e.g., null-effect or exception raised].
6.4J	01	The CAIS shall provide facilities to disable echoing of data units to their source.
6.4J	02	The CAIS shall provide facilities to enable echoing of data units to their source.
6.4J	03	The CAIS shall provide facilities to indicate whether echo-suppression is supported in the given implementation.
6.4J	04	The CAIS shall define the results of invoking the facilities to disable echoing in those implementations that do not support echo-suppression [e.g., null effect or exception raised].
6.4J	05	The CAIS shall define the results of invoking the facilities to enable echoing in those implementations that do not support echo-suppression [e.g., null effect or exception raised].

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)

Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 6 RAC Statements (cont.)
6.4K	01	The CAIS shall provide facilities to designate an input datastream as a control input datastream.
6.4L	01	The CAIS shall provide the ability to abort a process by means of trapping a specific data unit in a control input datastream of that process.
6.4L	02	The CAIS shall provide the ability to abort a process by means of trapping a specific data block in a control input datastream of that process.
6.4M	01	The CAIS shall provide facilities to specify the data unit that may be trapped.
6.4M	02	The CAIS shall provide facilities to query the data unit that may be trapped.
6.4M	03	The CAIS shall provide facilities to specify the data block that may be trapped.
6.4M	04	The CAIS shall provide facilities to query the data block that may be trapped.
6.4M	05	The CAIS shall provide facilities to disable the trap sequence at execution time.
6.4M	06	The CAIS shall provide facilities to enable the trap sequence at execution time.
6.4N	01	The CAIS shall support facilities for the dynamic control of data links [self-test].
6.4N	02	The CAIS shall support facilities for the dynamic control of data links [automatic dialing].
6.4N	03	The CAIS shall support facilities for the dynamic control of data links [hang-up].
6.4N	04	The CAIS shall support facilities for the dynamic control of data links [broken-link handling].

[ ] is used for criterion or to amplify the requirement.

Appendix A (cont.)
Complete List of RAC Requirements with Requirement Numbers

RAC Para	Rqmnt No.	Section 6 RAC Statements (cont.)
6.5A	01	The CAIS shall provide facilities for the specification of the size of a sequence of units during program execution.
6.6A	01	The CAIS shall specify a representation on physical media of a set of related data entities [referred to as the Common External Form].
6.6B	01	The CAIS shall provide facilities using the Common External Form to support the transfer among CAIS implementations of sets of related data entities such that attributes are preserved.
6.6B	02	The CAIS shall provide facilities using the Common External Form to support the transfer among CAIS implementations of sets of related data entities such that relationships are preserved.
6.7A	01	The CAIS shall cause only the task requesting a synchronous input operation to await completion.
6.7A	02	The CAIS shall cause only the task requesting a synchronous output operation to await completion.
6.7B	01	The CAIS should provide facilities to control the consequences when the physical device does not have all the of the features of the virtual device.

[ ] is used for criterion or to amplify the requirement.

APPENDIX B: Key Issues Study Statistical Details

KEY ISSUE #1

ADA

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
2.1	01	2	B	1/2/9
2.7	01	4	B	1
2.7	02	4	B	1
2.7	03	4	B	1
2.7	04	4	B	1
3.1A	01	1	B	1
3.1A	02	1	B	1
3.1B	01	4	B	1
3.1B	02	4	C	1
3.1C	01	4	B	1
3.1C	02	4	B	1
3.1C	03	4	B	1
3.1C	04	4	B	1
3.1C	05	4	B	1
3.2B	01	2	C	1/2
3.2B	02	1	C	1/2
3.2B	03	4	C	1/2
3.2C	01	1	B	1
3.2C	02	1	B	1
3.2C	03	1	B	1
3.2D	01	4	C	1
4.4E	01	2	B	1/4/5
4.4E	02	2	B	1/4/5
5.5B	01	1	C	1/2/4/6

Key Issue #1 is reflected in 24 of the RAC statements

75.0% (18/24) are Substantially Met (B)
25.0% (6/24) are Partially Met (C)

APPENDIX B: Key Issues Study (cont.)

KEY ISSUE #2

WIDELY IMPLEMENTABLE

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
2.1	01	2	B	1/2/9
2.3	01	1	B	2
2.3	02	1	B	2
2 3	03	1	B	2
2 3	04	1	B	2
3 1D	01	4	B	2
3.2B	01	2	C	1/2
3.2B	02	1	C	1/2
3.2B	03	4	C	1/2
3.2F	01	1	B	2
5.5B	01	1	C	1/2/4/6

Key Issue #2 is reflected in 11 of the RAC statements

63 6% (7/11) are Substantially Met (B)
36.4% (4/11) are Partially Met (C)

APPENDIX B: Key Issues Study (cont.)

KEY ISSUE #3

SECURITY

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
2.8	01	2	B	3
2.8	02	1	B	3
2.8	03	3	B	3
2.8	04	1	B	3
4.4G	01	2	B	3/5
4.4G	02	2	B	3/5
4.4G	03	2	B	3/5
6.3C	01	2	B	3/4/5/6
6.3C	02	2	B	3/4/5/6

Key Issue #3 is reflected in 9 of the RAC statements

100.0% (9/ 9) are Substantially Met (B)

APPENDIX B: Key Issues Study (cont.)

KEY ISSUE #4

INPUT/OUTPUT

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
4.4E	01	2	B	1/4/5
4.4E	02	2	B	1/4/5
5.5B	01	1	C	1/2/4/6
6.1A	01	2	B	4
6.1B	01	2	B	4
6.1C	01	2	G	4
6.1C	02	2	G	4
6.1D	01	2	G	4
6.1E	01	3	G	4
6.1F	01	3	G	4
6.2A	01	2	B	4
6.2B	01	2	B	4
6.3A	01	2	B	4
6.3B	01	2	B	4
6.3B	02	2	B	4
6.3C	01	2	B	3/4/5/6
6.3C	02	2	B	3/4/5/6
6.3D	01	2	C	4
6.3D	02	2	C	4
6.3D	03	2	C	4
6.3D	04	2	C	4
6.3E	01	2	G	4
6.3E	02	2	G	4
6.3F	01	2	B	4
6.3F	02	2	B	4
6.3G	01	2	G	4
6.3H	01	2	C	4
6.3H	02	2	C	4
6.4A	01	2	G	4
6.4A	02	2	G	4
6.4A	03	2	G	4
6.4B	01	2	C	4
6.4B	02	2	C	4
6.4C	01	2	G	4
6.4C	02	2	G	4
6.4D	01	1	G	4
6.4D	02	1	G	4
6.4D	03	2	G	4
6.4D	04	2	G	4
6.4E	01	1	G	4
6.4E	02	1	G	4
6.4E	03	2	G	4
6.4E	04	2	G	4
6.4F	01	1	G	4
6.4F	02	1	G	4

APPENDIX B: Key Issues Study (cont.)

KEY ISSUE #4 (Cont')

INPUT/OUTPUT

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
6.4F	03	2	G	4
6.4F	04	2	G	4
6.4G	01	2	G	4
6.4H	01	3	G	4
6.4I	01	2	G	4
6.4I	02	2	G	4
6.4I	03	2	G	4
6.4I	04	1	G	4
6.4I	05	1	G	4
6.4J	01	2	B	4
6.4J	02	2	B	4
6.4J	03	2	B	4
6.4J	04	1	B	4
6.4J	05	1	B	4
6.4K	01	2	G	4
6.4L	01	2	G	4
6.4L	02	2	G	4
6.4M	01	2	G	4
6.4M	02	2	G	4
6.4M	03	2	G	4
6.4M	04	2	G	4
6.4M	05	2	G	4
6.4M	06	2	G	4
6.4N	01	3	G	4
6.4N	02	3	G	4
6.4N	03	3	G	4
6.4N	04	3	G	4
6.5A	01	2	G	4
6.6A	01	1	G	4
6.6B	01	2	G	4
6.6B	02	2	G	4
6.7A	01	1	C	4/6
6.7A	02	1	C	4/6
6.7B	01	4	C	4

Key Issue #4 is reflected in 79 of the RAC statements

22.8% (18/79) are Substantially Met (B)
15.2% (12/79) are Partially Met (C)
62.0% (49/79) are Deferred Items (G)

APPENDIX B: Key Issues Study (cont.)

KEY ISSUE #5

ENTITY MANAGEMENT SUPPORT

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
4.1A	01	2	B	5
4.1A	02	2	B	5
4.1A	03	2	B	5
4.1B	01	2	C	5
4.1C	01	1	B	5
4.2A	01	1	C	5
4.2B	01	1	B	5
4.2B	02	1	B	5
4.2B	03	1	B	5
4.2B	04	1	B	5
4.2B	05	1	B	5
4.2B	06	1	G	5
4.2B	07	1	B	5
4.2B	08	1	G	5
4.2C	01	2	G	5
4.2C	02	2	G	5
4.2C	03	2	G	5
4.2D	01	2	G	5
4.2D	02	1	G	5
4.2E	01	2	A	5
4.2E	02	2	A	5
4.3A	01	2	A	5/7
4.3A	02	3	A	5/7
4.3A	03	1	A	5/7
4.3A	04	3	A	5/7
4.3B	01	2	B	5
4.3B	02	2	B	5
4.3B	03	2	B	5
4.3B	04	2	A	5/7
4 3B	05	3	A	5/7
4.3C	01	2	A	5
4 3C	02	2	A	5
4.4A	01	2	B	5
4 4A	02	2	B	5
4 4A	03	2	B	5
4.4A	04	2	B	5
4.4A	05	2	C	5
4.4B	01	2	B	5
4.4B	02	2	B	5
4.4B	03	2	B	5

4.4B	04	2	B	5
4.4B	05	2	C	5
4.4C	01	2	B	5
4.4C	02	2	B	5
4.4D	01	2	A	5/7
4.4D	02	2	A	5/7
4.4E	01	2	B	1/4/5
4.4E	02	2	B	1/4/5
4.4F	01	2	B	5/6
4.4F	02	2	B	5/6
4.4F	03	2	B	5/6
4.4G	01	2	B	3/5
4.4G	02	2	B	3/5
4.4G	03	2	B	3/5
4.5A	01	3	G	5/6
4.5B	01	3	G	5/6
4.5B	02	3	G	5/6
4.5B	03	3	G	5/6
4.5C	01	1	G	5/6
4.6A	01	3	B	5
4.6A	02	2	B	5
4.6B	01	3	C	5
4.7A	01	3	C	5
4.7A	02	3	C	5
4.7A	03	3	C	5
6.3C	01	2	B	3/4/5/6
6.3C	02	2	B	3/4/5/6

Key Issue #5 is reflected in 67 of the RAC statements

17.9% (12/67) are Not In CAIS (A)
52.2% (35/67) are Substantially Met (B)
12.0% (8/67) are Partially Met (C)
17.9% (12/67) are Deferred Items (G)

APPENDIX B: Key Issues Study (cont.)

KEY ISSUE #6

PROGRAM EXECUTION FACILITIES

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
4.4F	01	2	B	5/6
4.4F	02	2	B	5/6
4.4F	03	2	B	5/6
4.5A	01	3	G	5/6
4.5B	01	3	G	5/6
4.5B	02	3	G	5/6
4.5B	03	3	G	5/6
4.5C	01	1	G	5/6
5.1A	01	2	B	6
5.1B	01	2	B	6
5.1C	01	2	B	6
5.1D	01	2	B	6
5.1E	01	2	B	6
5.2A	01	2	B	6
5.2A	02	2	B	6
5.2B	01	3	B	6
5.2C	01	2	B	6
5.2C	02	2	B	6
5.2C	03	1	B	6
5.3A	01	2	B	6
5.4A	01	3	C	6
5.4B	01	2	B	6
5.4C	01	2	B	6
5.4D	01	2	B	6
5.4E	01	2	B	6
5.5A	01	2	B	6
5.5B	01	1	C	1/2/4/6
5.5C	01	2	A	6
5.5C	02	2	A	6
6.3C	01	2	B	3/4/5/6
6.3C	02	2	B	3/4/5/6
6.7A	01	1	C	4/6
6.7A	02	1	C	4/6

Key Issue #6 is reflected in 33 of the RAC statements

6.1%	(2/33)	are Not In CAIS	(A)
66.7%	(22/33)	are Substantially Met	(B)
12.1%	(4/33)	are Partially Met	(C)
15.1%	(5/33)	are Deferred Items	(G)

APPENDIX B Key Issues Study (cont.)

KEY ISSUE #7

Distribution

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
4.3A	01	2	A	5/7
4.3A	02	3	A	5/7
4.3A	03	1	A	5/7
4.3A	04	3	A	5/7
4.3B	04	2	A	5/7
4.3B	05	3	A	5/7
4.4D	01	2	A	5/7
4.4D	02	2	A	5/7

Key Issue #7 is reflected in 8 of the RAC statements

100.0% (8/ 8) are Not In CAIS (A)

APPENDIX B. Key Issues Study (cont.)

KEY ISSUE #8

LEVEL OF THE INTERFACES

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
2.2	01	4	B	8/9
2.4	01	4	B	8
3.2E	01	4	B	8

Key Issue #8 is reflected in 3 of the RAC statements

100.0% (3/ 3) are Substantially Met (B)

APPENDIX B: Key Issues Study (cont.)

KEY ISSUE #9

APPROPRIATENESS OF THE INTERFACES

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
2.1	01	2	B	1/2/9
2.2	01	4	B	8/9
2 3	05	1	C	9/10
2 3	06	4	B	9/10

Key Issue #9 is reflected in 4 of the RAC statements

75.0% (3/ 4) are Substantially Met (B)
25.0% (1/ 4) are Partially Met (C)

APPENDIX B. Key Issues Study (cont.)

KEY ISSUE #10

TECHNOLOGY COMPATIBILITY & EVOLUTION

RAC paragraph number	Rqmnt number	Action verb	Fulfillment code	Issue number
2.3	05	1	C	9/10
2.3	06	4	B	9/10
2.5	01	4	B	10
2.6	01	1	B	10

Key Issue #10 is reflected in 4 of the RAC statements

75.0% (3/ 4) are Substantially Met (B)
25.0% (1/ 4) are Partially Met (C)

To: KIT/KITIA committee members

From: Tom Atwood
Director of R&D
Ontologic Inc.
(617) 667-2383

Date: 09/18/86

Subj: Ontologic Object Manager and KAPSE dbms requirements

The attached note compares the Object Manager under development at Ontologic with the requirements for 'Entity Management Support' outlined in section 4 of the document entitled *Rationale for the DoD Requirements and Design Criteria for the Common APSE Interface SET (CAIS)*, dated 13 September 1985.

In brief, the Object Manager supports an object-oriented data model which meets all of the requirements outlined in the *Rationale*; it is a strict superset of the ER model chosen as a vehicle for discussing those requirements in the *Rationale*. Its principal extensions over the ER model are that it provides stronger notions of encapsulation and inheritance by supporting the definition of type-specific operations on user-defined types as well as the simple Get_attribute_value, Set_attribute_value, ... operations provided by the ER model on its built-in types.

I caution that this note is not a stand-alone document. It has to be read against the *Rationale* document. As in the *Rationale* I have carried forward in italics the RAC requirements. My own comments are in a roman font. The paragraphs in the *Rationale* which expand on or explain the RAC requirements are, as a general rule, not repeated. They are referenced by paragraph number. For each requirement, I have included a simple "OK", "NO", or "IN PART" in brackets. "OK" means we meet or exceed the requirement.

This note also rashly presupposes a basic understanding of the data model defined by the Object Manager. It has not been my objective to explain the OM data model; but simply to compare it with the RAC requirements. I have attached a separate paper which provides at least a high level overview of the OM data model.

The first production release of the Object Manager is scheduled for late next year; the system will be available in Beta-test form six months prior to that. We will have invested by that time over 60 man-years in its development. The release-2 system is expected to top 100 man-years.

We would be interested in working with the Committee to define an Ada package interface to the Object Manager, and/or with Ada compiler vendors who would like to consider incorporating it as the dbms component of an Ada KAPSE.

Please contact me or my research assistant, Jennifer Hertz, at (617) 667-2383 if you would like to discuss either of these options in more detail.

4. Entity Management Support

The general capabilities of the model specified by ihe CAIS are the following. The entity-relationship model, for which definitions and requirements follow in 4.1 .. 4.7, provides these capabilities and any alternative model of CAIS requirements must also.

The Object model is a strict superset of the ER model. Like the ER model, it supports entities, properties and relationships (we call them objects, attributes and relationships). Two principal advantages of the model per-se:

1. It includes type-specific operations.
2. It makes specification/implementation distinction

The advantages of the Object Model are: first, modularity and protection; second, performance. Coupled with type-specific operations, the Object Model allows the ct nstruction of highly efficient storage/retrieval/operations on objects which don't fit well into the record-oriented structures typically used to implement ER models: source-code; tree-structured annotated grammars; graph representations of high-level designs; graphs of the module structure of large systems, documentation, bit-maps used to capture screen images, graphical representations of project plans, etc.

There are significant advantages of the Ontologic implementation. The OM is directly targetted for Engineering Design to begin with - it is not an attempt to 'use' a system developed initially for business oriented problems. The data model has version/alternative support built in at low level where efficient. OM provides 'version consistency' rather than global consistency based on serializability. The system provides constraints w/triggers. Type managers for object-oriented graphics (2D and 3D) in addition to device/independent window/menu managers are built in features of the Ontologic system.

a. *There shall be a means for retaining data.*

   [OK] The Object Manager is a DBMS. Entities may persist beyond the life of the process which created them.

b. *There shall be a way of retaining relationships among and properties of data.*

   [OK].

   (paragraph 3). [OK]: This requirement is satisfied by ConsistsOf/APartOf relationships. It is not restricted to hierarchy and may be a graph.

   (paragraph 4). [OK]: OM supports multiple types of data, not just record-oriented data: text, source-text, documentation (with defined structure), graphs etc.

   (paragraph 5). [OK]: A motivating force for object model versus older relational models was that it "support a natural expression of the data that closely models the user's understanding of the problem that [s]he is working on". It can represent "the objects that the user is interested in", "the relationships among these objects" and "the dependencies between these objects".

c. *There shall be a way of operating upon data, deleting data, and creating new data.*

[OK]. OM is actually stronger than the ER model proposed due to the inclusion of type-specific operations.

d. *There shall be a means for defining certain operations and conditions as legal, for enforcing the definitions and for accepting additional definitions of legality.*

[OK] There exists a set of operations defined on built-in types (ENTITY, OBJECT, PROPERTY, ATTRIBUTE, RELATIONSHIP, OPERATION, ...) and the ability for schema designer to restrict use of domain-specific types to operations which make sense for those types.

e. *There shall be a means to describe data, and there shall be a means to operate upon such descriptions. Descriptions of the data shall be distinguished from the data described.*

[OK]. Schema is defined using an interactive Type Definition Language (TDL). A type defines the properties (attributes and relationships) carried by instances of the types, and the operations available on those instances. The types so defined are themselves stored in the dbms as objects, and may therefore be interrogated either interactively or programmatically just like any other objects in the system. OM offers sophisticated set of facilities for handling the evolution of the database schema.

(paragraph 3). [OK]: OM contains type specific operations. Note that this is missing from earlier semantic models (the ER model among them).

(paragraph 4). [OK]: a type may export several different interfaces which permit more or less power to users with access to that interface. Distinguishing access rights of individuals, projects, and company is straightforward.

(paragraph 5). [OK]: OM permits Access Control and provides security mechanisms.

(paragraph 6). [OK]: OM accomodates version/alternatives, percolation. Version consistency is base for Configuration Management subsystems which enforce policy.

f. *There shall be a way to develop new data descriptions by inheriting (some of) the properties of existing data descriptions.*

[OK]. A-kind-of (i.e., subtype/supertype) relationships defined on types. A subtype inherits operations and properties defined on its supertype.

g. *The relationships and properties of data shall be separate from the existence of the data.*

[OK]. The existence of an individual is independent of the values of any properties [s]he has, or relationships [s]he may participate in. Existence constraints can, however be defined which make the continued existence of an object dependent on its participation in particular relationships. Objects are identified by a unique-id (uid) which is independent of any property values.

h. *The descriptions and the instances of data shall be separate from the tools which operate on them.*

[OK].

The following characterization (subsections 4.1-4.7) of Entity Management Support (EMS) is based on the STONEMAN requirements for a database, using a model based on the entity-relationship concept. **Although a CAIS design meeting these requirements is expected to demonstrate the characteristics and capabilities reflected here, it is not necessary that such a design directly employ this entity-relationship model.** *The entity-relationship model for which definitions and requirements follow in 4.1 - 4.7 fulfills these requirements (a-h above), and any alternative data model shall fulfill these requirements and shall also fulfill the equivalent of the requirements in subsections 4.1 through 4.7.*

The Object Model on which the OM is built is a strict superset of the ER Model. It supports objects (entities) with attributes and relationships. But it also supports operations. This encapsulation of operations with data is at the heart of the notions of modularity and abstract data types which underlay the Ada notion of a package. We think the Ada KAPSE should live up to the standard set by Ada: if the ADT approach has advantages for temporary data (types and instances which do not survive the execution of the process), then it should have the same advantages for persistant data (data which outlives the process which created it); even more so: persistent data is often shared among several programs. Encapsulating it so that programs can operate on it only through operations which 'make sense' for that type of data, becomes all the more important in insuring the integrity and long-term maintainability of subsystems involving many programs.

4.1 Entities, Relationships, and Attributes

The introductory material in this section contains definitions of several terms. I repeat them here, and note the analagous OM term.

ENTITY: A representation of a person, place, event, or thing.

We call these OBJECTS.

15.110entities. A relationship among N entities (not necessarily distinct) is known an an "N-ary" relationship.

We use the same term RELATIONSHIP. In Release-1 we support only binary relationships, not n-ary relationships.

ATTRIBUTE: An association of an entity or relationship with an elementary value.

We use the same term ATTRIBUTE. Note that the requirement that relationships carry attributes as well as entities. We support that. Objects can have attributes. Relationships can have attributes. More formally, both object types and relationship types define attributes for which their instances carry values.

In the OM model as in the ER model, attributes take as their values *ELEMENTARY VALUES* -- things like integers, strings, etc. We call these Universals. We support integers, string, real, and enumeration data, as well as a rich set of aggregates of such data -- sets, bags, lists, vectors, matrices, etc. *UNITERPRETED DATA* would be handled as either a bit_vector, or a byte_vector.

4.1A Data

The EMS shall provide facilities for representing data using entities, attributes, or binary relationships. The EMS may provide facilities for more general N-ary relationships, but it is not required to do so.

[OK]. We support objects, attributes and binary relationships. The release-1 model does not support N-ary relationships.

(paragraph 1). [OK]

(paragraph 2). [OK]

(paragraph 3). [OK]

(paragraph 4). [OK]: Deleting an object deletes all of the attributes that belong to it. Similiarly, relationships carried by that entity cease to exist.

(paragraphs 5,6). These paragraphs suggest that some entities, "source files, test results, cross references, and schedules" may have "contents" in addition to attributes and relationships. This is contrasted, in paragraph 6, with the RAC which "uses the concept of "attribute" top represent all values: the source text of a program is held as an attribute whose elementary value may be uninterpreted data". We follow the RAC model in handling "uninterpreted data", but would suggest that for the examples chosen -- source files, cross references, schedules, etc. -- the object model allows the definition of efficient representations for these types which make their structure semantically visible. A source-file need not be a block of "uninterpreted data" to be made sense of only by sub-rosa agreement between the text editor, compiler, and cross-reference tools. On the contrary, the structure of the source file -- perhaps both its physical decomposition into objects like lines, pages, characters, and its logical decomposition into packages, declarations, bodies, statements, etc. -- can and should be defined within the model. In an ER model this is often not done for the pragmatic reason that implementations of the model often support only a single representation for all types -- records with fields, or perhaps a tree of nodes where each node contains slots -- and this representation is simply not efficient for non record-oriented data. Ontologic's Object Manager allows the type definer to define not only a **Specification** for a type, but, if he wants, to also define a custom **Implementation** for instances of that type. An Implementation consists of a **Representation** and a set of **Methods** which implement the Operations defined in the Specification in terms of the underlying type-specific Representation.

4.1B Elementary Values

The EMS shall provide facilities for representing data as elementary values.

[OK]. The OM supports integers, reals, characters, strings, booleans, enumerated types (e.g., type GRAIN instances wheat, corn, barley; cargo:GRAIN), unnamed enumerated types (e.g., cargo: oneof(wheat, corn)), and aggregates made up on these (e.g., MATRIX of INTEGER).

(paragraph 1). This paragraph contains a good example of the kind of awkwardness which occurs using a model (e.g. ER) which does not support operations. "Also an entity could be the representation of an I/O device or running process which accepts input as uninterpreted data written to one (or more) of its attributes or produces output to be read from an attribute." It would be semantically cleaner to define the operations Read and Write on the type IO_DEVICE than to try to treat this as setting and getting attribute values. Note the awkwardness of trying to handle the invocation of an operation which has several arguments, as setting all at once the values of several attributes. Note also that 'forcing' the ER model this way leaves you without any good way of handling Exceptions raised by what is really an Operation. Things are a lot simpler -- there is less 'forcing' -- if the data model allows the type programmer to define operations on abstract types. Again, the fundamental point we are making is that there is no reason that persistent abstract data types should be treated differently from transient ones.

4.1C System Integrity

The EMS facilities shall ensure the integrity of the EMS-managed data.

[OK]. The OM supports recovery from unexpected failure -- process crash, system crash, disk failure, network partition. It supports a notion of nested atomic actions. This mechanism is used not only to provide recovery in the event of failures invisible to the program, but also to allow a program to extend to its user the ability to 'abort' a transaction in progress when [s]he discovers that [s]he is changing something [s]he did not intend to be changed. The user does not have to go back and undo by hand each change he made.

d. There shall be a means for defining certain operations and conditions as legal, for enforcing the definitions and for accepting additional definitions of legality.

[OK]. The OM allows the definition of Constraints on the legal values of attributes and relationships. Further, it allows the definition of Operations which are legal on instances of a type, along with constraints on the type of the arguments these operations take and the results they return. The model also allows the type programmer to specify the Invariants which the set of operations defined on the type must obey. The system does not attempt to 'prove' the invariants, but it does give the type programmer a way of formally stating intent -- something which we think may be important over the long-term maintenance and evolution cyle.

e.There shall be a means to describe data, and there shall be a means to operate on such descriptions.

[OK]. Type Definition Language is used to describe data. Those descriptions can be queried or modified either interactively through the Object Editor or programmatically through the normal DML interface exported to programs. Type definitions are themselves objects.

(paragraphs 3-5). We take a DBMS-like approach based on a notion of transactions, rather than an operating sytem (OS)-like approach based on building sufficient redundancy into data structures to permit (partial) reconstruction in the even of failure.

Attributes take as their value universals -- things like numbers, strings, names, etc. Note that universals are immutable, indistinguishable (one "3" is hard to tell from another), and can often be represented by small indices. This has the important consequence that universals can be cached directly in slots of the representation of the object that refers to them. Relationships, on the other hand, since they refer to particulars which are mutable (and often have complex representations in and of themselves), have to be implemented as pointers to the representation of the particular to which they refer.

(paragraph 3). [OK]

4.5 Typing

The following definition, used in this subsection, pertains to the remainder of section 4 also.

TYPING An organization of entities, relationships, and attributes in which they are partitioned into sets, culled entity types, relationship types, and attribute types, according to designated type definitions.

The OM distinguishes types and instances. We use the term "type", and make a formal distinction between the intentional notion of type, and the "class" which is its extensional analogue. The world of types is partitioned as follows:

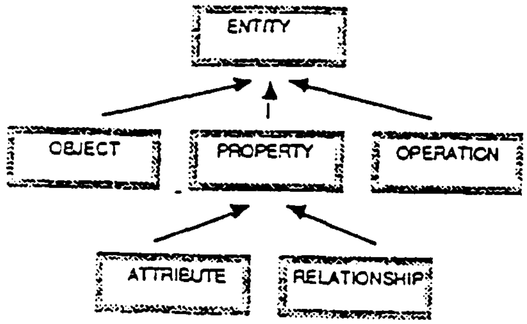


Figure 1. Root of the type lattice.

The arrows represent the subtype/supertype relationships which obtain between types, e.g, ATTRIBUTE is a subtype of PROPERTY, which is in turn a subtype of ENTITY.

4.2A Types

The facilities provided by the EMS shall enforce typing by providing that all operations conform to the type definitions. Every entity, relationship, and attribute shall have one and only one type.

[OK]. All entities have one immediate type. However the type hierarchy is a lattice. A type may have multiple supertypes, but a single individual cannot be of more than one immediate type.

(paragraph 3: *At the time we wr< 'e the RAC, there appeared to be two models: (1) every object has exactly one type and types are arranged in a lattice or (2) objects could be of several types. We concluded that the requirements we wanted to express could be expressed in the two models in equivalent ways, but that it was easier to express them in the first model; thus this requirement.)*

The OM has also taken the first approach.

4.2B Rules about Type Definitions

The EMS type definitions shall

- o *specify the entity types and relationship types to which each attribute type may apply.* ([IN PART]: However we found it too cumbersome to use independent declarations of attribute and relationship types, stating explicitly for each, which object types it applied to. It proved much more natural for people to define the object types and state which attributes and relationships they carry. Given that a subtype inherits attributes and relationships from its supertype(s), there proved to be very little redundance of attribute declaration, and a great improvement in clarity.)
- o *specify the type or types of entities that each relationship type may connect and the attribute types allowed for each relationship type.* ([OK]; however, see comment above.)
- o *specify the set of allowable elementary values for each attribute type.* [OK]
- o *specify the relationship types and attribute types for each entity type.* [OK]
- o *permit relationship types that represent either functional mappings (one-to-one or many-to-one) or relational mappings (one-to-many or many-to-many).* [OK]
- o *permit multiple distinct relationships among the same entities.* [OK]
- o *impose a lattice structure on the types which include inheritance of attributes, attribute value ranges (possibly restricted), relationships, and allowed operations.* [OK]

(paragraph 1) Noted.

(paragraph 2) [OK] The OM supports "strong" subtyping, i.e., in contrast to the Smalltalk, "behaves like except" model of subtyping. Strong subtyping allows compiler optimization and integrity assertions which the weak model does not.

(paragraphs 3-6) [OK] This is just a restatement of the need for multiple supertypes.

4.2C Type Definition

The EMS shall provide facilities for defining new entity, relationship, and attribute types.

[OK] This can be done either interactively, using the Schema Editor, or by a program, using the DML interface.

4.2D Changing Type Definitions

The EMS shall provide facilities for changing type definitions. These facilities must be controlled such that data integrity is maintained. [OK] Release-2 will go further than this, allowing the database schema not only to evolve, but to do so without requiring changes to existing programs which see the database through an older schema. This is one area of research where we feel we are significantly ahead of not only the commercial market, but most of the research community as well.

(paragraph 1). [OK]

(paragraph 2). [OK] The central theme of Release-2 is supporting evolution, both in the structure of the programs stored in the database and in the structure of the database schema.

(paragraph 3: *one may want to modify existing types that are known to a set of tools without modifying the tools that use the type).* [OK] Handled with versions of types (Release-2).

(paragraphs 4-5). [NO] Retrospective verification that existing instances obey constraints added later is not available in Release-1. Release-2 contains a general purpose constraint verification subsystem which handles a very wide class of problems like this.

4.2E Triggering

The EMS shall provide a conditional triggering mechanism so that prespecified procedures or operations (such as special validation techniques employing multiple attribute value checking) may be invoked whenever values of indicated attributes change. The EMS shall provide facilities for defining such triggers and the operations or procedures which are to be invoked.

[OK] OM provides more general implementation, both in terms of (a) when the trigger can be invoked, and (b) what it can do. Triggers can be attached not only to getting or setting attribute values, but to the invocation of any operation, e.g., recompiling one package body, might trigger relinking of the subsystem containing that package. Similarly the 'operations or procedures' which the trigger can execute are not limited to those built into the ER model -- e.g., 'get/set attribute value', 'define/delete relationship' -- but may include as well abstract Operations defined by the application builder on object types unique to the designated domain. The 'Link' operation triggered in the scenario above is a good example. Rich support for daemons and triggers is another example of the advantage of having a full object-oriented model -- one which includes operations -- rather than a more restrictive ER model.

4.11 Identification

4.3A Exact Identities

The EMS shall provide exact identities for all entities. The EMS shall support exact identities for all relationships. The exact identity shall be unique within an instance of a CAIS implementation, and the EMS shall support a mechanism for the utilization of exact identities across all CAIS implementations.

[IN PART]. All objects are identified by UIDs which are unique across a local area network. We are looking into supporting optional uid prefixes which would make these ids unique across a wide-area-network and/or unique period. We have not committed to this yet. We may judge the overhead to be too high. The general form of a UID is:

```
[organization].[WAN].[LAN].entity_id.[version_id].[copy_id]
   where
entity_id ::= [area].[segment].[chunk]
```

4.3B Identification

The EMS shall provide identification of all entities, attributes and relationships. The EMS shall provide identification of all entities by their exact identity.

[IN PART] Objects, attributes, and relationships may all be identified by name. Only Objects, however, have UIDs or what the RAC calls "exact identities".

(paragraph 2). This paragraph seems to call for something slightly weaker than the RAC. It would permit relationships as well as attributes to have only user names, not exact identities. This is in fact, what the OM does. Internally we use UIDs for relationships which carry attributes; we will also use them for n-ary relationships -- i.e., in situations when you want to reference the relationship itself, rather than simply cross it to the object(s) to which it refers.

4.3C Identification Methods

The EMS shall provide identification of entities and relationships by at least the following methods:

- o *identification of some "start" entity(s), the specification of some relationship type and the specification of some predicate involving attributes or attribute types associated with that relationship type or with some entity type. This method shall identify those entities which are related to the identified start entity(s) by relationships of the given relationship type and for which the predicate is true. Subject to the security constraints of section 2.8, all relationships and entities shall be capable of identification via this method, and all attributes and attribute types (except uninterpreted data) shall be permitted in the predicates.* [OK] o *identification of an entity type or relationship type and specification of some predicate on the value of any attribute of the entity type or relationship type. This method shall identify thos entities or relationships of the given type for which the predicate is true. Subject to the security constrains of section 2.8, all attributes (except uniterpreted data) shall be permitted in the predicate.* [OK].

The OM supports **instance identifying expressions** such as:

PROFESSOR[name='Weihl'].advisees[status=Phd_candidate].

Evaluating this expression would return a set containing those advisees of Professor Weihl who were Phd candidates.

4.15 Operations

The OM supports not only operations defined on the types built into the system -- ENTITY, ATTRIBUTE, and RELATIONSHIP -- but operations defined by users, on domain-specific types, e.g., a 'Compile' operation on the type SOURCE-CODE.

4.4A Entity Operations

The EMS shall provide facilities to:

o *create entities*

o *delete entities*

o *examine entities (by examining their attributes and relationships)*

o *modify entities (by modifying their attributes)*

o *identify entities (as specified in Section 4.3)*

[OK]

4.4B Relationship Operations

The EMS shall provide facilities to:

o *create relationships*

o *delete relationships*

o *examine relationships (by examining thier attributes)*

o *modify relationships (by modifying their attributes)*

o *identify relationships (as specified in Section 4.3)*

[OK]

(paragraph 1). This again underscores the importance of having a robust notion of Operations in the model. Triggers which fire when specific operations are executed (e.g., 'Compile'), may be used to set the values of attributes defined on the object(s) which the operation touches (e.g., 'time_last_compiled').

4.4C Attribute Operations

The EMS shall provide facilities to:

o *examine attributes*

o *modify attributes*

[OK].

4.4D Exact Identity Operations.

The EMS shall provide facilities to:

o pass exact identifiers between processes

o compare exact identities

[OK]

4.4E Uninterpreted Data Operations

The EMS shall provide that use of the input-output facilities of the Ada language results in reading/writing an uniterpreted data attribute of an entity. The facilities of Section 6 shall then apply.

[No] Modifying the Ada runtime to work with the Object Manager is something which we would have to undertake in concert with an Ada compiler development house.

4.4F Sychronization

The EMS shall provide dynamic access synchronization mechanisms to individual entities, relationships and attributes.

[OK]

4.22 Transaction.

4.5A Transaction Mechanism.

The EMS shall support a transaction mechanism. The effect of running transactions concurrently shall be as if the concurrent transaction were run serially.

[OK]

4.5B Tran_action Control.

The EMS shall support facilitie: to start, end and abort transactions. When a transaction is aborted, all effects of the designated sequence of operations shall be as if the sequence were never started.

[OK]

4.5C System Failure.

System failure while a transaction is in progress shall cause the effects of the designated sequence of operations to be as if the sequence were never started.

[OK]

4.26 History

4.3A History Mechanism

The EMS shall support a mechanism for collecti.,g and utilizing histories. The history mechanism shall provide sufficient information to support comprehensive configuration control

[OK] Release-2 notions of versions, alternatives, percolation of versions up the APO hierarchy are designed to act as a high level platform for domain specific configuratio.i control subsystems.

4.3B History Integrity

The EMS shall support mechanisms for ensuring the fidelity of the history.

[OK]

4.29 Robustness and Restoration

4.7A Robustness and Restoration

The EMS shall provide facilities which ensure the robustness of an ability to restore EMS-managed data. The facilities shall include at least those required to supp : t the backup and archiving capabilities provided by modern operating systems.

[OK]. Initial release runs on single Workstation. Release 3 will run fully distributed in a LAN containing heterogeneous workstations and mainframes/megaminis.

AN OBJECT-ORIENTED DBMS FOR DESIGN SUPPORT APPLICATIONS

Thomas M. Atwood

Ontologic Inc., Billerica, MA 01821

ABSTRACT

The database requirements of engineering design support applications are in some ways significantly different from those of general business applications. We discuss a database management system, the Object Manager, which has been built to address this new set of issues. It provides support for abstractions generally found only in AI knowledge representation systems: a-kind-of (AKO), a-part-of (APO), and an-instance-of (AIO). It also provides support for tracking versions and alternatives. The latter is applicable to both individual designs and also to the schema of the database itself.

INTRODUCTION

The Ontologic Object Manager (OM) is a database management system targetted explicitly at design support applications (E-CAD, M-CAD, AEC, etc.). It is designed to run in a distributed fashion in a network of high performance engineering workstations and database servers.

In this paper we focus on two problems central to design environments:

- design complexity, and
- design evolution.

We outline some of the features of the Object Manager which address these problems. Section 2 discusses the abstraction mechanisms defined by the data model. Section 3 discusses version and alternative support.

DESIGN COMPLEXITY

Design support applications are some of the most complex artifacts of human endeavor. They represent tens of man years of development effort and often hundreds of thousands of lines of code. At the core of their complexity is the fact that they attempt to support one of the activities of the human mind most difficult to reduce to rule and number—creation: in this case design. Engineering design is the process of building up a model of a complex artifact. The artifact itself may be decomposed into hundreds of other components. And at any level of the decomposition there may be several logical ways of looking at the design as well as several alternatives for its physical realization.

The first requirement of a design support database management system is a data model which is strong enough to bring order to this complexity. The way people normally deal with complexity is through abstraction. Conventional commercial database management systems support only one type of abstraction—*instantiation*. In a Codasyl system a record is an instance of a record type; a particular set is an instance of a set type. In a Relational system, a tuple is an instance of the type defined by its containing relation.

AI knowledge representation systems have for many years incorporated two other basic abstractions:

- *generalization* (variously referred to as 'a-kind-of' or 'is-a'), and
- *aggregation* ('a-part-of' or 'consists-of')

The Object Manager supports all three types of abstraction. We discuss each briefly below.

Instantiation (An-Instance-Of)

Our fundamental modeling construct is the *ENTITY*. The class of all entities is partitioned along two orthogonal lines into:

1. *types* and *instances*
2. *objects*, *properties*, and *operations*.

We say that entities are *instances* of *types*. Types define the properties carried by each of their instances and the operations which may be executed on each instance.

The system comes with a very general set of built-in types which define a framework for modeling the objects of more specific application domains. The OEM, or end-user builder of an engineering design support application, programs the system by defining a set of types which model the entities of interest in his particular application. As we will see in the next section, these types are defined as *subtypes* of the built-in types, and so they automatically *inherit* the basic structuring behavior defined by the built-in types.

The built-in types at the root of the type lattice are the following:

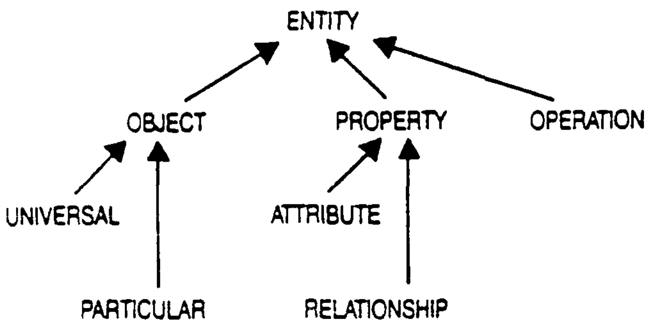


Figure 1-1

Our notation convention is that up-arrows model a-kind-of relationships; the horizontal line indicates that the subtypes *OBJECT, PROPERTY,* and *OPERATION* partition the set of entities: any entity must be either an object, a property, or an operation

Database models have historically focused only on properties; they provide only a generic set of operations (*Join, Project, Select*) which operate on the containers (*records* and *tuples*) which are the only constructs the database provides for modeling real world objects. Programming languages have had the opposite bias; everything is modeled as an operation. Programming-language-based systems such as Smalltalk [1] or Flavors [2] tend to force the type definer to specify operation triples (*get_foo, set_foo,* and *init_foo*) to handle what are really properties. The OM model includes **both** properties and operations.

Properties in the Object Manager may be set-valued or single-valued. They take as their values abstract objects, not just strings or integers as in conventional data models.

We distinguish properties as either *attributes* or *relationships* on the basis of the kind of objects which they take as values. Attributes take *universals* as values, e.g., the *age* attribute defined on *PROFESSOR* may, for a particular professor, take as its value the number '32'. Things like numbers, strings, colors, etc. we call *universals*. This is in contrast to tangible objects which occupy specific positions in space and time. We refer to the latter as *particulars*. Attributes take universals as their values. Relationships take particulars as their values. The *advisees* relationship defined on *PROFESSOR*, for example, takes as its value a set of students. Students are particulars. This distinction between universals and particulars, whatever its philosophical interest has pragmatic value for the Object Manager implementation. Since particulars are mutable they must have a single stored representation to which each object that references them refers. If the particular changes, its new state should be seen by all of the objects which refer to it. Universals, on the other hand, are immutable, and can therefore be freely replicated. Since universals are often elements of ordered sets, they can be represented as indices. These indices occupy about the same amount of space as an inter-object reference and are much smaller than a full-blown stored object. The Object Manager takes advantage of these facts by caching the representation of universals in the stored representation of the objects which refer to them.

Relationships may be *distributive*, or *collective*. For example, the type *PROFESSOR* might define either a single *advises* relationship which takes as its value a set of students; alternatively *PROFESSOR* might define a set of *advisee* relationships, each of which takes as its value a single student. The first is the most succinct if you are interested in simply which students the professor advises. The second is more powerful. It permits the user to record information about each particular *advises* relationship, e.g., when it started, whether it has worked satisfactorily, etc. Note that this is not information about either the professor or the student, but about the relationship between them.

The need to model properties of relationships led to two other features of the model. First, it is **entities** which have properties and operations, not objects. Since *PROPERTY* is a kind of *ENTITY* and all entities can have properties, then properties can have properties. Second, instances of a type do not simply have values for each of the properties defined by the type: they actually carry specific instances of the properties themselves. For example, if the type *MEMORY_CHIP* defined the property *implementation_technology*, then each individual memory chip (formally each instance of the type *MEMORY_CHIP*) not only carries a particular value for that property, e.g., 'NMOS', 'CMOS', 'HMOS', etc., but actually also carries a particular instance of the property type. This property instance may itself have properties.

The ability for a relationship to have properties is used to effect a generalization and consolidation of the model when compared to systems like Smalltalk. Inheritance is no longer a special purpose mechanism hardwired into the interpreter for a restricted set of relationships (i.e., AKO). It can now be modeled as a property of a relationship. It can therefore be attached to any relationship, not just those built into the interpreter. This allows the application builder to define inheritance across relationships unique to his application domain, and can again contribute to simplifying the design problem.

Generalization (A-Kind-Of)
Types are related to one another in subtype/supertype hierarchies. For example, *ASSISTANT_PROFESSOR* is a type of *PROFESSOR*. Given a pair of hierarchically related types, the more general one is termed the *supertype*, the more specific one the *subtype*. In our example *PROFESSOR* is the supertype of *ASSISTANT_PROFESSOR*; *ASSISTANT_PROFESSOR* is a subtype of *PROFESSOR*. What gives this notion its power is that it serves as the base for an automatic inheritance mechanism. If *ASSISTANT_PROFESSOR* is a subtype of *PROFESSOR*, which is in turn a subtype of *PERSON*, then *PROFESSOR* inherits all of the properties defined on *PERSON*, and *ASSISTANT_PROFESSOR* inherits all of those defined on both *PERSON* and *PROFESSOR*. The database designer need not go through the drudgery of defining common properties on each subtype in the hierarchy. This reduces the specification task and also allows a significant increase in modularity of representation.

Multiple Supertypes and Specialization

First generation object base models restricted inheritance to a strict hierarchy of types (i.e., a type could only have one supertype). As experience using these systems accumulated, demand emerged for the more general inheritance incident to a lattice or partial order of types. Examples are the Traits mechanism added to Mesa by Xerox [3] and the Flavors mechanism added to Lisp by Symbolics [4]. Figure 2-1 illustrates such a type lattice. *INSTRUCTOR* is a type with multiple supertypes.

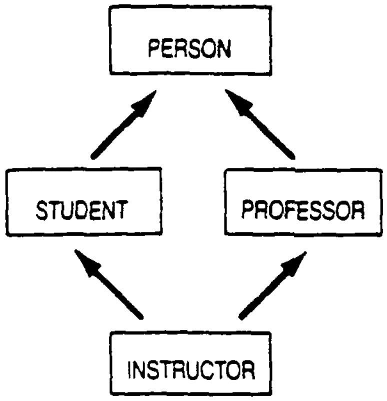


Figure 2-1

The notion of multiple supertypes carries with it two implementation problems:

- name conflict between properties inherited from different supertypes, and
- interaction between inheritance from different supertypes.

The first problem has straightforward solutions and is not of particular interest. The second has led us to a new, more powerful notion of specialization.

For a type with a single supertype, inheritance can be strictly defined. If *A* is a kind of *B*, then *A* inherits all of the properties and behavior of *B*. When a type has two or more supertypes, it is often inadequate to say that it simply inherits all of the properties of all of its supertypes. The inheritance from different supertypes interacts. In the example of Figure 2-1, instructors are like students in that they are enrolled in degree programs. However, they often do not take any courses. Instructors are like professors in that they teach courses. However, unlike professors, they do not have advisees. This kind of cross-type inheritance blocking cannot be handled even by pushing the choice as to which supertype dominates the other down to the level of individual properties. *INSTRUCTOR* blocks some properties of *STUDENT*, but not by virtue of a conflicting inheritance from *PROFESSOR*. It might be possible to say that an implicit property of *PROFESSOR* (that professors don't take courses) has blocked an explicit property in *STUDENT*, but this seems to be stretching it a bit. It is not clear how an interpreter should intuit which of the properties not present on *PROFESSOR* should block properties actually present on *STUDENT*.

In Figure 2-2 we illustrate a different way of handling multiple inheritance.

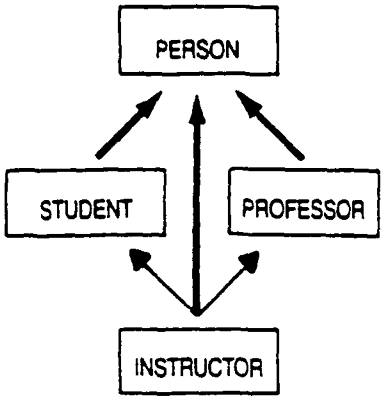


Figure 2-2

A specialization has a single *base type* (represented by a heavy arrow) and one or more *specializers* (represented by the light arrows.) In figure 2-2 we say that *INSTRUCTOR* is a subtype of *PERSON*, specialized by *STUDENT* and *PROFESSOR*. Inheritance across the base type arrow is by default strong. Everything which is true of a *PERSON* is, unless explicitly blocked, also assumed true of an *INSTRUCTOR*. Inheritance across the specializer arrow(s) is more idiosyncratic. By default nothing is inherited. As we alluded to above, this is controlled by defining a property on the relationship.

Partitions

The notion of specialization interacts strongly with the notion of *partitioning*, mentioned briefly above. A type *X* defines a set of instances {x}. This set may be partitioned into several disjoint sets {xi},{xj},... {xm} on the basis of some property of *X* which has a limited number of distinct values. An example might be the type *SHIP* with the partition defining attribute *engine_type* which can take the values 'steam-turbine', 'diesel', or 'nuclear'. If each of the partitions is defined in the schema as a subtype then we get something like Figure 2-3:

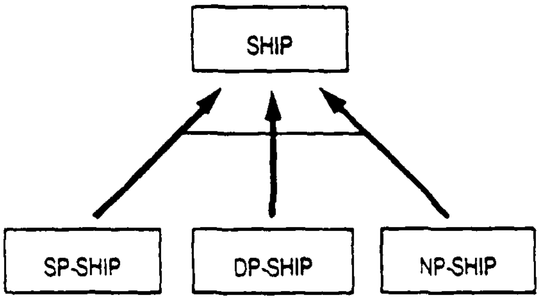


Figure 2-3

The horizontal line indicates that the subtypes are mutually disjunct. Disjunct means that anything which is an instance of one subtype cannot also be an instance of another. The integration of the notion of specialization with the notion of partition-defining attributes gives us a powerful and simple way of enforcing some basic semantic constraints, and lets us extend the same facilities to the builders of design support applications.

Partitions can also be defined without explicit incarnation as subtypes. An attribute of a type can be identified as a *partition-defining attribute*. Each value which the attribute takes implicitly defines a partition member. It is up to the database designer whether some, all, or none of these partitions are actually made into separate subtypes. They may serve as the referent for specializer arrows whether they are types or attribute values. For example, we could say that *NUCLEAR-POWERED-SHIP* represents a specialization of *SHIP* BY < *engine_type* = 'nuclear'>.

If there are two independent partition-defining attributes in the base type, it is possible to generate subtypes representing the cross product of the values of the two partitions:

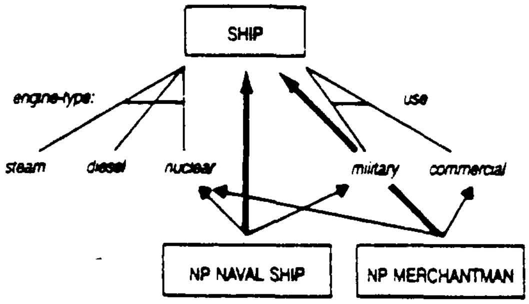


Figure 2-4

If we are interested in the cross product of one value of one partition-defining attribute with several values of a second partition-defining attribute, we can handle this by instantiating the first attribute value as a subtype and then using this as the base type for two further specializations:

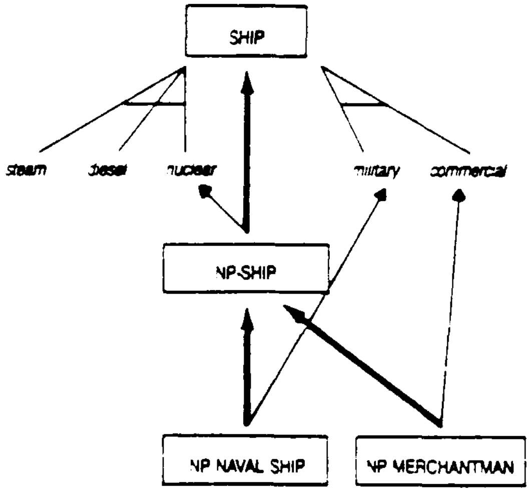


Figure 2-5

This gives us a place to hang operations which are specific to nuclear powered ships, and could therefore not be specified in the type *SHIP*, but don't need to be repeated in the definition of *NP_NAVAL_SHIP* and *NP_MERCHANTMAN*. In general, the choice of whether to use a value of a partition-defining attribute as a full-fledged subtype depends on whether the type programmer has 'more to say' about the partition member. Its use as a constraint, though, is independent of whether it is an attribute or a set of subtypes.

In the common case where a type has only a single supertype, the specialization model we have presented here reduces to the classical supertype model. To keep the diagrams simple, when the specializer and the base-type arrow refer to the same type, we do not require a separate specializer arrow. This lets the database designer keep his diagrams simple, when simple diagrams are sufficient to convey his meaning. But it still gives him the power of the specialization notion when the latter helps to bring order to an intrinsically complex problem domain.

In sum, we have replaced the notion of uncoordinated inheritance from multiple supertypes, with a notion of coordinated inheritance based on *specialization*.

Individuals with Multiple Immediate Types

In section 2.2.1 we discussed a type having multiple supertypes. In the Object Manager an individual can also be of multiple types. This introduces a distinction between an *instance* and an *individual*. In models like TAXIS [5] an individual may be of multiple types, but these types are all hierarchically related. He has only one *immediate type*. The other types in which he participates are all supertypes of this one immediate type. In the Object Manager an individual can have more than one immediate type. A particular *PERSON*, for example, might be both a *STUDENT* and a *PROFESSOR*.

To sharpen the contrast between an individual who is an instance of two types and a type which has two supertypes, consider a university database schema which defines the types *PERSON, STUDENT, PROFESSOR* and *INSTRUCTOR*. *STUDENT* and *PROFESSOR* are subtypes of *PERSON*. *INSTRUCTOR* is a subtype of both *STUDENT* and PROFESSOR. Like students, instructors are enrolled in a degree program. Like professors, they teach classes. Consider now the contrast between the way we would model two individuals, Jeff and Sam. Jeff is finishing his PhD at MIT and has been given a stipend as a instructor. Sam is a full professor at MIT, but is taking a course in linguistics at Harvard. We would represent Jeff as an individual who is an instance of the single type *INSTRUCTOR*. We would represent Sam as an individual with two roles—one as an instance of the type *PROFESSOR*, and the other as an instance of the type *STUDENT*. Note that Sam is clearly **not** an *INSTRUCTOR*.

The notion of partitions introduced in section 2.2.2 above provides a constraint on the types of which an individual can be an immediate instance. These types cannot be elements of a disjunct partition (or subtypes of such elements). An individual cannot for instance be both a *PROFESSOR* and a *ROCK* because *PROFESSOR* is a subtype of *PERSON* and *ROCK* is a subtype of *THING*, and *PERSON* and *THING* are elements of a disjunct partition defined on *OBJECT*.

Aggregation (A-Part-Of)
The APO abstraction is ubiquitous in design support applications. Things are made up of other things. Design is in fact often a process of specifying the subcomponent structure of a complex artifact.

Two elements of the Object Manager's support for APO abstraction are worth highlighting:

1. The APO abstraction is not limited to defining a hierarchy between objects; it can support a lattice.
2. There is built-in support for property inheritance both up and down the hierarchy

The ability to handle lattices as well as hierarchies is important in design applications which aggregate objects in more than one way. Good examples are the logical/physical aggregation hierarchies we illustrate below for technical publishing and computer-aided design:

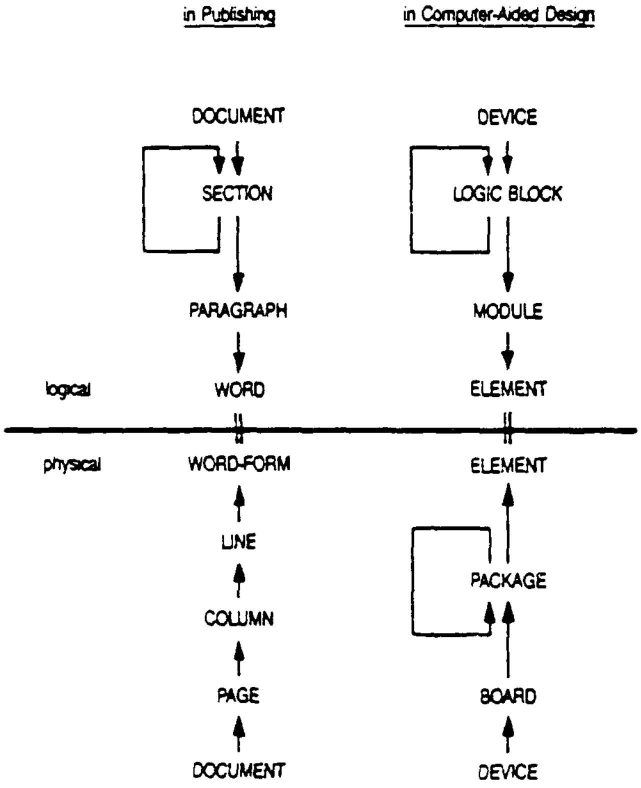


Figure 2-6

Note that in each case the hierarchies touch at their lowest levels. This is characteristic of a wide class of design support applications. In systems which support only strictly hierarchical aggregation, the lowest-level components have to be replicated to allow aggregation into different hierarchies. In the Object Manager they can share low-level components without forcing replication of data. This can dramatically reduce storage requirements and removes from the application all of the burden of keeping replicated copies of the same data coordinated.

A second feature which can reduce storage requirements, and improve performance for systems involving small objects, is the Object Manager's support for type-specific property value inheritance down the APO hierarchy. In the engineering documentation example shown in Figure 2-6 above, each character of a document was treated as a separate object. An individual character may have 4–8 typographic properties—font, size, face, kerning, etc.—most of which are actually determined by a specification on a much higher-level object in the logical hierarchy—SECTION_BODY for example. If values for these properties can be 'inherited' by an object at a lower level in the hierarchy, then they need not be physically stored as part of the representation of each and every character. A performance-related side effect of this notion of property value inheritance down an APO hierarchy, is that it is straightforward to refine the *get_property_value*, and *set_property_value* operations on the types in the APO hierarchy to cache current property values in a 'state machine' from which they can be very rapidly retrieved without even going to the underlying database. We have taken advantage of this possibility in our own implementation of the graphics subsystem. The graphics processor caches *editing contexts* which hold the current attribute values determining things such as the width and style of lines to be drawn in special-purpose local memories on the GPU board which can be read at very high speed by the GPU microcode.

DESIGN EVOLUTION

Commercial database management systems are what we term "shadow" systems. They track the real world. Since the real world has one current state, the database has one current state. That makes sense in the commercial applications these systems were designed to handle. If you are booking reservations on a 767 to San Francisco, that plane actually has 184 seats. It does not do any good to consider a theoretical airplane with 204 seats. Twenty people are going to be left at the gate.

In design support applications, however, it is precisely these versions and theoretical alternatives that you want to track.

The Object Manager provides built-in support for both versions and alternatives.

The notion of versions is orthogonal to the distinction between types and instances. Formally it is modeled as a third partition of the class ENTITIES:

1. *type / instance*
2. *object / property / operation*
3. *history_bearing_entity / non_history_bearing_entity*

The current implementation handles only the cross products involving OBJECT; it does support, however, both history-bearing types and history-bearing instances. We limit our discussion in this paper to versions of instances.

Versions
Figure 3-1 shows a type (CPU_CHIP) and a particular instance of that type (the NSC 32332) in a notation which should be familiar from our introduction to objects in section 2 above.

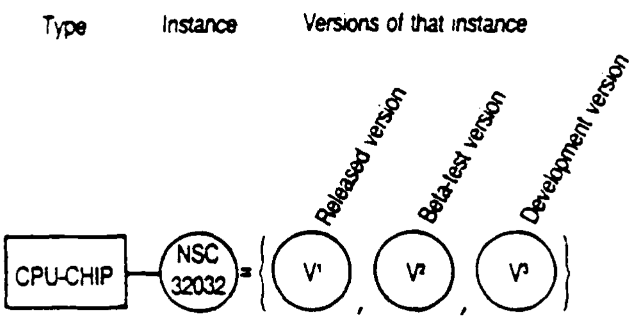


Figure 3-1

The Object Manager does not actually store representations for instances per se. An instance in the Object Manager is actually modeled as a set of distinct *versions*. In our example there is a 'released' version, a 'beta-test' version, and a version 'under development'. Individual versions of the same conceptual object may have important differences in property values—in our chip example, such things as the range of temperature over which the version is stable, known problems, clock speed, etc..

To return to the comparison with today's commercial database management systems, the real database in a commercial system is the audit log; what is termed the database is really just a cache containing the most recent version of each object in the database. The problem with attempting to use these systems as a foundation for design support applications, is that there is no way for the programs which implement the application to get at older versions of objects. In a typical commercial dbms or access method, an application program identifies an object either by a *key value*, or more generally by an *associative retrieval expression*. In either case, though, what the database returns when it is given this reference is the most recent version of the object. This makes it impossible to get at an older version, because the data manipulation language simply gives the application programmer no way of asking for one. This makes it difficult for the design support application to extend to its users the ability to compare and contrast successive versions of an object or a mutually consistent set of objects. About the only option the DSA programmer has is to 'circumvent' the system—to roll the whole database back to a point in time at which the version he wants was extant; copy the version(s) he wants out of the database to an operating system file; then roll the database forward to its current state, and compare the version in the file with the version in the database. At best, then, the programmer is forced to step outside of the system to do what he wants. At worst, the time involved in rolling the database back and then rolling it forward again is so prohibitive that he either builds his own version-naming scheme on top of the database, or simply does not provide this functionality to his end user.

The Object Manager, by contrast, automatically tracks the evolution of an object through successive versions, and lets the user address them using an object-id of the form:

<conceptual object id>[<version id>]

The <version id> is optional. If it is not supplied, the system returns the most recent version by default.

The most recent version is normally stored in a fully articulated form: older versions are stored using a backward differential representation. This results in very low storage overhead for older versions.

The programmer/user has control over when a set of property value changes is considered significant enough to constitute a new version of an object, and this same control extends to how changing one object affects related objects.

Alternatives

The version path shown in Figure 3-1 was linear; each version had a single predecessor and a single successor. Complex designs often evolve in more complex ways. The Object Manager handles this by allowing version paths to fork into *alternatives*. Alternatives can be individually addressed using the version-id scheme introduced earlier. They may therefore be individually modified, compared and contrasted in a straightforward fashion.

Interaction with Aggregation

The notion of versions and alternatives interacts with the notion of aggregation to provide a powerful facility for managing the evolution of complex designs. In this section we discuss *percolation, contexts, partial consistency*, and the sharing of common components by alternative designs.

Percolation

When a new version of a lower-level component in an APO hierarchy is created, the designer may want this to automatically trigger the creation a new version of the higher-level aggregate object. We call this *percolation*. Figure 3-2 shows a workstation consisting of a bus, a memory subsystem, a CPU and a GPU. The initial version of the workstation, W1.v1, consisted of the initial version of each of these components: B1.v1, M1.v1, C1.v1, and G1.v1 respectively. A new version of the GPU, G1.v2 has been created which has percolated up the APO hierarchy to create a new version of the workstation. Note that the new version of the workstation shared the single extant version of the CPU with its predecessor.

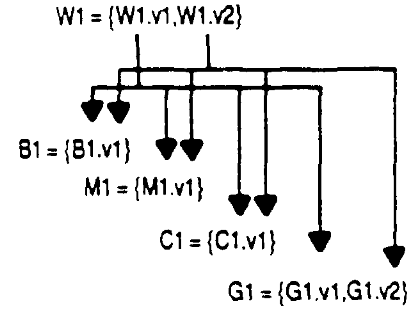


Figure 3-2

Since there are cases when the designer does not want a small change to a low level component to ripple through an entire hierarchy, creating a new version of everything above it, the model allows the type definer to control when and how far up the APO hierarchy version percolation occurs. If an object type O contains a property P which takes as its value an object of type V, P may be declared either *version-sensitive*, or *identity-sensitive* to V. If it is version-sensitive, then for any given instance of O, say On, the creation of a new version of the instance of V to which P refers will cause the creation of a new version of On. If P is declared identity-sensitive, then a new version of On will be created only when P is updated to refer to a different object, not just a different version of the same object. If P is declared neither identity-sensitive nor version-sensitive, then it is by default *insensitive*, and no change to the value of P will trigger the creation of a new version of On. Since the APO hierarchy is based on a relation, *(consists-of)* and since relations are properties, the sensitivity mechanism can be used to control the percolation of new versions to higher-level aggregate objects.

Contexts

So far we have a means of identifying versions of individual objects, but no way of grouping a set of coordinated changes to several objects which make sense together. Early research prototypes which attempted to support design evolution [6,7,8,9] handled this problem by introducing additional concepts into their data model. In PIE [8], for example, *layers* grouped sets of related changes, and *contexts* were defined as a sequence of layers.

In the Object Manager individual versions of high-level objects are used to group mutually consistent sets of related objects, and each alternative path within the version history of an object serves a function analagous to PIE's contexts. Figure 3-3 illustrates this by taking the workstation example introduced in Figure 3-2 a step further.

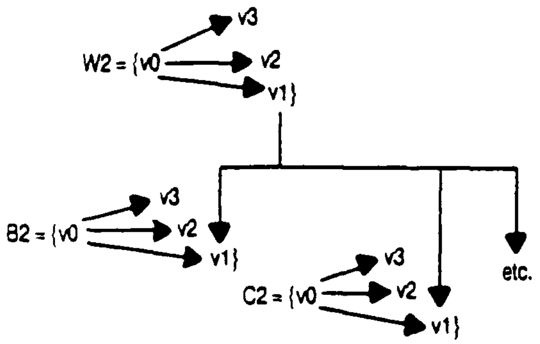


Figure 3-3

Assume that the initial workstation design used a proprietary synchronous bus speed-matched to the fastest CPU chips available at the time. For the second implementation, the designers decide to evaluate two alternatives: (1) a higher speed gate-array based implementation of the proprietary bus, (2) an asynchronous, industry standard bus for which VLSI bus interface chips are commercially available. They want to consider two variations on the second alternative, one based on a VME bus, the other based on a Multibus-II. They cannot make an informed choice until all three alternatives have been elaborated in sufficient detail to understand their cost, and to permit simulation of their performance. The arrow shown in Figure 3-3 represents a consistent set of components for one version of one of these alternatives. Note that it is nothing other than an instance of the APO relationship, in particular the one defined on version W2.v1 of the workstation. It serves the function of defining an internally consistent 'context' for object references. Although there are times when programs want to compare alternatives explicitly, much of the time they work within the context of a single alternative, and would prefer not to have to deal with the complexity of identifying each version of object explicitly. The Object Manager handles this by allowing the program to specify a particular version of an aggregate object as defining a naming context within which to resolve all object references not otherwise explicitly qualified.

Note that since the aggregation hierarchy is potentially many levels deep, this gives us a tree-structured context mechanism without adding any new constructs to the model.

Partial Consistency

Concurrency control in business database management systems is based on the notion of global consistency. A *transaction* takes the database from one globally consistent state to another. In a design database the notion of global consistency is not useful. A design database may not achieve a globally consistent state for weeks or months; in fact, it may never do so over the period that it is useful in supporting the design process. The point at which it achieves consistency is by definition the point at which the design is complete. At any particular point in its evolution, specific versions of specific portions of the database may be consistent with other portions of the design, but no more.

Using versions of aggregate objects to group mutually consistent versions of lower level components gives us a simple handle on what is consistent with what at any given point in time. Since the APO hierarchy can be many levels deep, and since changes to lower level components need not always percolate to the top of the hierarchy, the Object Manager gives design support systems a succinct way of modeling the fact that different portions of the design may be internally self-consistent without the whole design necessarily being consistent.

The fact that the APO relationship does not define a strict hierarchy, but rather a lattice, can also be of interest in modeling partial consistency. High-level aggregate objects are often decomposed, in the design process, along two or more different lines. The logical/physical decomposition shown in Figure 2-6 is common in several disciplines. If information on the consistency of a design is attached to the different *consists_of* properties which model these different decompositions, it is possible to track the achievement of design consistency along one dimension independent of its achievement along others. It is common for a design to reach an initial state of logical consistency, for example, before all of the constraints introduced by the problem of physical realization have been considered.

Sharing Common Components

In systems designed to track the evolution of designs for complex artifacts, the question of how to manage the evolution of the elements of the design which are common to two alternatives often looms as large as the problem of modeling the elements which differ. The workstation example discussed above is a case in point. The initial simulation models were done using a commercially available relational dbms. Given the lack of alternative support, the job was done by constructing three separate databases, one for each design. The result was an administrative nightmare, in trying to keep all three databases abreast of changes to the portions of the design which were common to two or more of them. One day a set of simulation runs showed that the VME bus was the best alternative. A day later, the same simulation showed that a higher-speed proprietary bus was the only real solution. The design engineers could simply never be sure that two ostensibly successive simulation runs were actually based on comparable data.

In the Object Manager, a high-level conceptual object like a system design is represented as the root node of an APO hierarchy which ties it to all of its subordinate module designs. Evolution of lower-level objects common to two alternative versions of the design is handled by simply having both versions of the design refer to the same versions of lower-level objects. Both designs will then automatically see the most recent versions of lower-level objects as the lat er are modified.

Conclusion

The Ontologic Object Manager has coupled the notion of versions and alternatives with an object-oriented view of data to produce a database management system which provides a strong foundation of engineering design support applications.

References

[1] A. Goldberg and D. Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley, 1983.

[2] D. Weinreb and D. Moon. *The LISP Machine Reference Manual,* 1982.

[3] G. Curry, L. Baer, D. Lipkie, and B. Lee, "Traits: An Approach to Multiple-Inheritance Subclassing".

[4] Weinreb, Ibid.

[5] J. Mylopoulos, P. Bernstein, and Wong, "A Language Facility for Designing Database-Intensive Applications", ACM Transactions on Database Systems, Vol 5, No. 2, June 1980, pp. 185–207.

[6] G. Sussman, and D. McDermott, "From PLANNER to CONNIVER —A genetic approach", *Fall Joint Computer Conference,* Montvale, NJ, AFIPS Press, 1972.

[7] G. Hendrix, "Expanding the Utility of Semantic Networks Through Partitioning", *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence,* 1975, pp. 115-121.

[8] D. Bobrow and I. Goldstein, "Representing Design Alternatives", *Proceedings of the Conference on Artificial Intelligence and the Simulation of Behavior,* Amsterdam, July 1980.

[9] R. Katz, "A Database Approach for Managing VLSI Design Data", Proceedings of the 19th IEEE Design Automation Conference, pp. 274-282, 1982.

CAIS Computer Security Considerations

COMPUSEC, Inc.

OVERVIEW

- CAIS Security Assumptions
- CAIS Supports Security Policy
- CAIS Security With Respect To KAPSE, MAPSE And APSE

CAIS SECURITY ASSUMPTIONS

- CAIS **Does Not** implement security policy
  - -- Security policy enforcement provided by security kernel
  - -- A process can access some object (notably the segment descriptors) On!y when executing in the kernel itself
  - -- Therefore, **At Least** two hierarchical domains must be supported (i.e., kernel and user)

CAIS SECURITY ASSUMPTIONS (cont.)

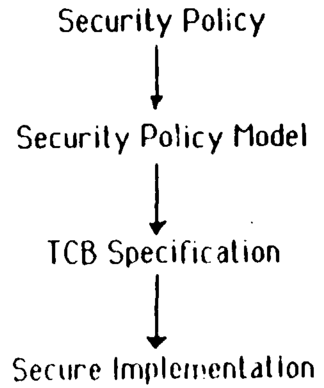
- CAIS resides between the kernel and user applications
- CAIS specifies the set of valid arguments and functions passed by the applications or user to the kernel
  - -- User sees CAIS when looking at kernel
  - -- Kernel sees CAIS when looking at user

CAIS SECURITY ASSUMPTIONS
(cont.)

- Examples of how CAIS would specify security
  - -- Input parameters
    (transparent to untrusted functions)
  - -- Output parameters
    (mediated by security kernel)
  - -- CAIS *provides well-defined entry* points into security kernel functions (procedure calls)

SECURITY DESIGN

- TCB is the totality of the mechanisms
  - -- Must be founded on a formal security policy model
- Formal security model is the mathematical representation of the protection policy enforced by the security kernel
  - -- The protection policy identifies all permissible modes of access between all subjects and objects
  - -- A given system is "secure" only with respect to some specific policy



SECURITY POLICY MUST SHOW THROUGH CAIS

CAIS will need to provide for the specific set of protection policies

-- Access control

TYPES OF ACCESS CONTROL

- Mandatory access control
- Discretionary access control
- Access control based on user role

MANDATORY ACCESS CONTROL

- Based on comparison of the user's clearance and the classification of the information being accessed
- *Comprised of*
  - -- Hierarchical levels (TS, S, C, U)
  - -- Non-hierarchical attributes (compartments)

DISCRETIONARY ACCESS CONTROL

- Limits authorization for access to objects to named users or user groups
- Establishment of access control is performed by the "owner" of the object, typically the creator of the object
- Restrictions based on
  - -- READ
  - -- WRITE
  - -- EXECUTE
  - -- DELETE

ACCESS CONTROL BASED ON USER ROLES

- Bounds the scope of the functional security access capabilities of users
- Some user roles
  - -- computer operators
  - -- maintenance personnel
  - -- security administrator
  - -- developmental personnel
- Note that a single user may have several valid roles, and may need to choose a specific access role at logon

EXAMPLE OF USER ROLE ACCESS CONTROL

- Security administrator can access:
  - -- user security attributes
  - -- segment table database
  - -- security audit trail
- No other user role has access to this data

EXAMPLE
READ ACCESS

- Subject security level >- object security level (MAC hierarchical)
- Non-hierarchical classification(s) of object is a subset of non-hierarchical clearance(s) of user (MAC non-hierarchical)
- User role must be sufficient to access object (user role enforcement)
- "owner" of object must have granted read access to user attempting to gain read access (DAC)

EXAMPLE
WRITE ACCESS

- Subject security level <= object security level (MAC hierarchical)
- Non-hierarchical clearance(s) of subject must is a subset of non-hierarchical classification(s) of object (MAC non-hierarchical)
- User role must be sufficient to write object (user role enforcement)
- "Owner" of object must have granted write access to user attempting to gain write access (DAC)

OTHER ACCESS CONTROL CONSIDERATIONS

- Security caveats

  -- NOFORN
  -- EYES ONLY

- Device environment

  -- port level - lesser of
  (user clearance, device location)

TRUSTED SUBJECTS

- Provides for more tailored security policy than defined in basic security policy model

- CAIS must provide kernel with set of interfaces that can be invoked only by trusted subjects

- Trusted subjects need to be recognized via some identifier, such as a privilege indicator

- If running program has such privileges, it can perform actions not normally permitted by the security policy model

TRUSTED SUBJECTS

- Usually perform system maintenance and control the access *policy that kernel* enforces

- Example

  Security policy model does not allow the lowering of the access class of information (tranquility property)

  Trusted subject could downgrade a data segment that a user accidentally overclassified

IMPLEMENTATION CONSIDERATIONS

- Hardware must support at least two hierarchical domains

  -- Kernel
  -- User
  -- (Supervisor recommended)

- Security kernel manages resources

  -- memory
  -- disk space
  -- multiple users

- Common utilities that do not manage anything shared among users reside outside kernel

- A process typically experiences a large number of kernel calls - *therefore a need* to ease the transfer of program control between domains exists

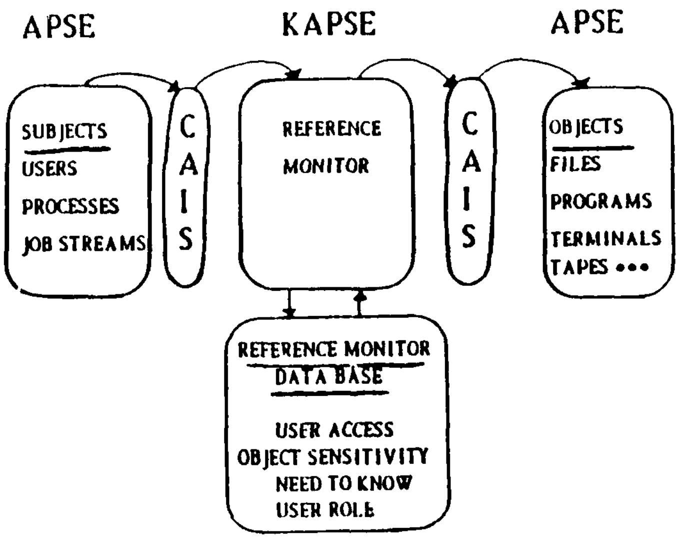
IMPLEMENTATION CONSIDERATIONS
(CONT.)

- CAIS can limit kernel entrance in two ways

  -- using a system fault or trap that tranfers control to known location

  -- Each kernel function has own entry point in a procedure call like fashion

CAIS SECURITY WITH RESPECT TO KAPSE, AND APSE


APSE
KAPSE
APSE
SUBJECTS
USERS
PROCESSES
JOB STREAMS
CAIS
REFERENCE MONITOR
CAIS
OBJECTS
FILES
PROGRAMS
TERMINALS
TAPES •••
REFERENCE MONITOR DATA BASE
USER ACCESS
OBJECT SENSITIVITY
NEED TO KNOW
USER ROLE